BSc (Hons) Artificial Intelligence and Data Science

Module: M2606 - Data Engineering

Individual Coursework Report

Module Leader: Mr. Mohamed Ayoob

RGU Student ID : 2410213

IIT Student ID : 20233168

Student Name :LOGANATHAN THUSHARKANTH

Contents

Table of figures

QUESTION 1 – OpenWeatherMap ETL TASK

1. Introduction

This project buillds an autometed **ETL** pipeliine using **Python, Apache Airflow, and AWS S3** to fetch real-time waether data from the **OpenWeatherMap API**, convert it into a structtured JSON object, and store it in **AWS S3** as part of a data lake. Apache Airflow orcheestrates the pipeliine to run every hour, offeriing smoth data flow and guarranteed scheduliing. It also provides hands-on trainiing in baic data enginering concepts such as ETL development, cloud storage, workfliw automation, and API integration, providing practical exposuree to cloued platforms and tols.

2. Setup & Configuration

2.1 Platform Selection: AWS EC2 (Ubuntu 22.04 LTS)

I deciided to ues an **AWS EC2** instace with **Ubuntu 22.04 LTS** (HVM, SSD Volume Type) for this projct due to compatibilitty isues with Ubuntu 24.04 and Apache Airflow. Ubuntu 22.04 LTS provides a stable and highely suported envrionment to run Airflow, making it more suitablle for producttion-level impllementation. On booting the instance, necesary ports were confiigured — **SSH, HTTP**, and a dedicated **TCP prot** for the Airflow web UI — to enable proper acess and communiication. (AWS, n.d.)

2.2 System Setup and Tool Installation

After launching the EC2 instance, the systtem was updated to include all the dependencies and securitypatches updated in order to have a godd fonudation to install and run Apache Airflow.

```
# Update system
sudo apt update && sudo apt upgrade -y
```

Figure 1

Next, Python 3, pip, and virtualenv were instaled to have a good Python environment in charge. Python 3 is the new standard of modern data enginering pipelines and is requiired for a good Airflow setup.

```
# Install Python and virtual env
sudo apt install python3-pip python3-venv -y
```

Figure 2

created a **Python virtual environment** (airflow_env) and activated it

```
# Create virtual environment
python3 -m venv airflow_env
source airflow_env/bin/activate
```

Figure 3

set specific versions for Apache Airflow and install Apache Airflow (Documentation, n.d.)

```
# Set versions
AIRFLOW_VERSION=2.8.1
PYTHON_VERSION=$(python --version | cut -d " " -f 2 | cut -d "." -f 1,2)
CONSTRAINT_URL="https://raw.githubusercontent.com/apache/airflow/constraints-${AIRFLOW_VERSION}/c

# Install Airflow
pip install "apache-airflow==${AIRFLOW_VERSION}" --constraint "${CONSTRAINT_URL}"
```

Figure 4

Airflow Configuration and Database Initialization

```
export AIRFLOW_HOME=~/airflow
airflow db init
```

Figure 5

Creating an Admin User for Airflow

```
airflow users create \
    --username admin \
    --firstname yourname \
    --lastname yourname \
    --role Admin \
    --email your@email.com \
    --password admin
```

Figure 6

Starting Airflow Services (Webserver and Scheduler)

```
# Start Airflow services
source airflow_env/bin/activate
airflow webserver --port 8080
source airflow_env/bin/activate
airflow scheduler
```

Figure 7

tested the Airflow setup by accessing the **Airflow UI** at the following URL

```
http://<your-ec2-public-ip>:8080
```

Figure 8

2.3 Creating an S3 Bucket and Configuring IAM Role

I createtd an **S3 bucket** (my-data-engineering-coursework) to store the waether data. **AWS S3** (Services, n.d.)was utiliized for its scalabillity and reliiability as a data lake storages. To access secuerly from the EC2 instance that is runing Airflow, an IAM role was createtd and atached and assiigned EC2Full Access and S3Full Access permissiions. (Services., n.d.)

2.4 Setup for OpenWeatherMap API

I regiistered on the OpenWeatherMap platfrom to obtain the API key, which is requiired to make authentticated requests to fetch real-time weather deta (OpenWeather, n.d.)

3. Building the ETL Pipeline

This Airflow DAG runs **hourly** and perfroms 4 steps: **Check** → **Extract** → **Transform** → **Load**. Here's what each tesk does:

3.1 API Availability Check – check_api

The HttpSensor is used to ping the OpenWeatherMap API befoer the ETL proces can be started. The **DAG executtes** (dags, n.d.)only when the API is accessiible and the response contaiins the keywrod "**main**." This approach prevents faiilures of pipellines by making the API operatiional before it goes ahead to extact anything.

```
check_api = HttpSensor(
    task_id="check_api_availability",
    http_conn_id="openweather_api",  # Defined in Airflow UI
    endpoint=f"data/2.5/weather?q={CITY}&appid={API_KEY}",
    method="GET",
    response_check=lambda response: "main" in response.text,
    poke_interval=10,
    timeout=60,
    dag=dag
)
```

Figure 9

3.2 Extract – extract_task

This task **fetches the current weather data for Portland** using the **OpenWeatherMap API** and saves the raw respoonse as a **timestamped JSON file** in the /tmp/ directory. It uses a **GET request** to retriieve the data and stres it in a fille named like portland_weather_raw_<timestamp>.json. This stap is crucial because it **captures real-time weather data** and maintais a **versioned record** that can be used for trackiing or auditing purpposes.

```
def extract(**kwargs):
    execution_date = kwargs['ts_nodash']
    url = f"https://api.openweathermap.org/data/2.5/weather?q={CITY}&appid={API_KEY}"
    response = requests.get(url)
    data = response.json()
    raw_filename = f"/tmp/{CITY.lower()}_weather_raw_{execution_date}.json"
    with open(raw_filename, "w") as f:
        json.dump(data, f)
```

Figure 10

3.3 Transform – transform_task

This tesk reads the rew **JSON** fille and extracts only the key fiields such as city, timestamp, temperature, humidity, waether description, and wind sped. The cleaed and formated data is then output to a new JSON fille in the /tmp/ directory. This makes the data wel formatted and ready for analyssis or storage, without redundant or noisy datta.

```python
def transform(**kwargs):
    execution_date = kwargs['ts_nodash']
    raw_filepath = f"/tmp/{CITY.lower()}_weather_raw_{execution_date}.json"
    with open(raw_filepath) as f:
        data = json.load(f)

    transformed = {
        "city": data.get("name"),
        "timestamp": data.get("dt"),
        "temperature": data.get("main", {}).get("temp"),
        "humidity": data.get("main", {}).get("humidity"),
        "weather": data.get("weather", [{}])[0].get("description"),
        "wind_speed": data.get("wind", {}).get("speed")
    }

    transformed_filepath = f"/tmp/{CITY.lower()}_weather_transformed_{execution_date}.json"
    with open(transformed_filepath, "w") as f:
        json.dump(transformed, f)
```

Figure 11

3.4 Load – load_task

This tesk **uploads the transformed data (in JSON format) to an S3 bucket** using the **boto3** library. The data is storred underr the patth waether-data/portland_weather_<timestamp>.json. This sttep ensures that the **processed waether data is securely stored in the cloud**, making it easiily accessiblle for **future analysis, reporting, or retrieval**.

```python
def load(**kwargs):
    execution_date = kwargs['ts_nodash']
    transformed_filepath = f"/tmp/{CITY.lower()}_weather_transformed_{execution_date}.json"
    s3_key = f"weather-data/{CITY.lower()}_weather_{execution_date}.json"

    s3 = boto3.client("s3")
    s3.upload_file(transformed_filepath, S3_BUCKET, s3_key)
```

Figure 12

4. Automation and Scheduling

4.1 Schedule Interval:

The DAG is set to run automaticaly every hour with **@hourly**, which means the tasks will execute once every hour, ensuriing that the data is allways up-to-date.

```
dag = DAG(
    dag_id="weather_etl_dag",
    default_args=default_args,
    start_date=datetime(2024, 1, 1),
    schedule_interval="@hourly",
    catchup=False
)
```

Figure 13

4.2 Airflow Retry Logic:

Airflow takes care of rettrying failed tasks by seting the retries parameter in the **default_args.** If a task faiils, it will automaticaly **retry once after a 2-minute delay**, preventing manuall intervention.

```
default_args = {
    "owner": "airflow",
    "retries": 1,
    "retry_delay": timedelta(minutes=2)
}
```

Figure 14

4.3 Logging Task Status:

Airflow automaticaly logs the status of each task, so you can monitor and track the progres of the ETL pipelline, ensuring that you are aware of any isues during executtion.

4.4 Execution Context:

The ***provide_context=True*** argument in the **PythonOperator** allows access to executtion metadata like **timestamps (ts_nodash)**. This is used to creatte timestamped filenames for rew and transforrmed data filles, ensuring each file is uniiquely identifiiable.

```
extract_task = PythonOperator(
    task_id="extract_weather_data",
    python_callable=extract,
    dag=dag,
    provide_context=True  # Access to ts_nodash, etc.
)
```

Figure 15

5. Evaluation and Testing

5.1 Testing Summary

1. **Manual Triggering and Scheduled Execution:**

   The DAG was manualy trigered from the Airflow UI, which alowed imediiate testting of task execution. This ensurred that the DAG could execte as expected when trigered manually.

   The DAG was also set to excute on schedulle by utillizing **@hourly**, meaning that it automatiically trigers every hour. This automated trigering ensured that the schedulling process was working well, executing tesks on expected intervals.

2. **File Saving in /tmp/ Directory:**

   After each DAG run, rew and transforrmed JSON filles were being saved in the /tmp/ directory. This indiicated that the extract and transform tesks were being executed successfuly, and the system was corectly handlling the fille creation after each task. The raw data was being extracted, and the transfromation logic was being appllied corectly, resullting in valid transformed filles being saved localy for further procesing.

3. **S3 Upload Failure and Solution:**

   The upload to **AWS S3** failled in the load tesk initialy. This was traced by vieving the Airflow logs and consolle output, which provided valluable deteils on what had failled.
   The error ocurred because there was a Boto3 initiallization error.

   After some round of troubleshoting, including reinstaling Boto3 in the Airflow virtuall enviromnent and reviewing the logs, the isue was traced and fixed. Once the isue was fixed, the uplload tesk could upload the transfromed filles to the target S3 bucket.

   Manuall verification was caried out by checking the S3 bucket through the AWS S3 Console, where the uploaded filles were confiirmed to be stored corectly. This final verifiication step ensured that the data had ben migrated and made accesible in the cloud.

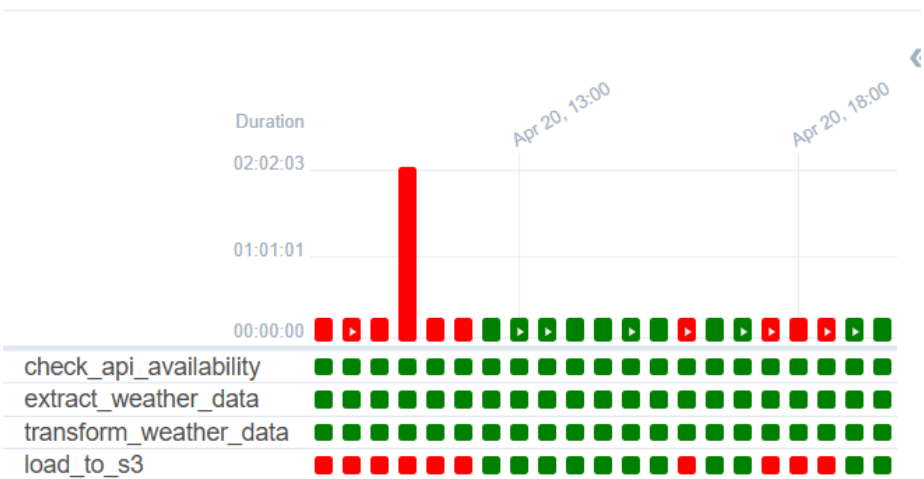

Figure 16

5.2 Challenges Faced & Solutions Applied

1. **Boto3 Module Error**
   Airflow failled to load the task due to an inport error with boto3. This was due to the fact that the boto3 modulle wasn't properly instaled within the Airflow enviroment (venv). This caused the tesk to faill when atempting to push the converted fille onto S3.
   To fix this, I re-instaled boto3 in the Airflow virtuall environment (venv) and ensured the instalation. Once that was done, the uplload task too worked as expected, and the fille uploaded sucessfully to

S3. The issue was fixed by ensuring the boto3 modulle had been installed corectly, and the upload porcess ran without isues.

```
pip install boto3
```

Figure 17

2. **File Overwrite Issue Due to Same Filenames**
   To strat with, I used **kwargs['ds']**, which retriieved only the date (e.g., 20250420) and hence provided the same fillename when the DAG was runs every time. This caused S3 to overwrite the previously uploaded fille because the names were the same.
   To fix this, I switched to utiizing **kwargs['ts_nodash']**, which includes the timestamp (like 202504201200). This adjjustment ensured that each DAG run produced a uniique file from a timestamped fille name so that no S3 files were clobered. As a result, all filles had a distinct name and uploaded succesfully without overwriting previious ones.

```
execution_date = kwargs['ts_nodash']
transformed_filepath = f"/tmp/{CITY.lower()}_weather_transformed_{execution_date}.json"
```

Figure 18

5.3 futhure improvements

1. Use S3Hook Instead of Raw Boto3

   S3Hook is the recomended way of working with S3 in Airflow. It simpliffies hooking into AWS, adds additiional security through the automated managment of credentials, and maintaiins the code simpler to write by bringing Airflow's connection management into it. Switching to S3Hook will secure and streamlline your fille uploads.

2. Use Logging Module Instead of print()

   Airflow works best when logs are procesed through the logging modulle. Using loging instead of print stataments means that logs can be captured in the Airflow UI, so you can simmply view tesk status and debug problems. This change will improve the robustnes of your error handlling and make the workfllow more professionall.

3. Store API Key in Airflow Variables

   Hardcodeing API keys is insecure. Storing API keys as Airflow Variablles alows you to centrallize and safely store sensitiive information. This reduces the likeli hood of exposing your credentials and ads a layer of security to your project as a wholle.

4. Utilize XCom to Share File Paths Between Tasks

   Sharing data, like fille paths, betwen tasks with the help of XCom makes your workflow more modullar and easiier to manage. It ensures your tasks are decouplled, reducing dependencies and meximizing your DAG's flexibiity. It also helps with task-speciffic data managament and enhances maintainabiity for your code.

5. Add a FileSensor Before Transform

   To prevent erors during data conversiion, the presence of a Fille Sensor will ensure that the raw fille exists before any conversiions are atempted. This operation will pervent tasks from running prematurely, ensuring that data is availablle for procesing when needed. It will also reduce the likeli hood of erors from mising or incomplete filles.

6. Concusion

This project showcases sucessfuly constructing and automatiing an end-to-end real-time data pipelline using **Airflow, Python, and AWS,** showcasing competence in API integration, data processing, and cloud storage using AWS S3. The pipelline performs data extraction, transfromation, and loading (ETL) very eficiently with neglligible human interventiion. Key takeaways are seamless integration with AWS services for safe cloud storage, automatic handlling of waether data, and Airflow for procesing intricate workfllows. The project can be exttended furthar by incorporating aditional cities or waether condittions, storing data in databases to quary, and integrating with dashboarrding platforms like **Amazon QuickSight** or **Graffana** for real-time analytics. Overal, this project gives a strong foundation for scallable, self-automating data pipellines that posess the future ability to grow and adapt.

7. Video Link

https://drive.google.com/file/d/1cNqBaZgcviXH04R707QccRQ1BtKFB6UM/view?usp=sharing

QUESTION 2 – SUPERMART

1. Identify Dimension Tables:

1. Store_Table

Attribute	**Description**
store_id (PK)	Unique identifier for each store.
store_name	Name of the store.
city	City where the store is located.
state	State where the store is located.

country	Country where the store is located.
region	Business/administrative region.

Role:

This dimensiion table contains the geogrraphical information of each stoer.
It suports analysisi by store locatiion (city, state, country) and can be utillized for the compariison of the salles performance of diferent regiions.

2. Product_Table

Attribute	Description
product_id (PK)	Unique identifier for each product.
product_name	Name of the product.
category	Category of the product (e.g., electronics, clothing).
brand	Brand of the product.
unit_price	Price of the product per unit.
supplier_id / supplier_Name	ID of the supplier for the product.

Role:

This table contaiins product-specifiic information, which hellps in product-wiise salles performance anallysis, price, inventory management, and suppllier details. It enablles detaiiled analysiis of salles by suplier, brand, and categorry.

3. Customer_Table

Attribute	Description
customer_id (PK)	Unique identifier for each customer.
customer_name	Full name of the customer.
address	address of the customer.

membership_level	Loyalty program level (bronze, silver, gold).

Role:

This table contaiins customar profiile data, which is improtant for loyallty analysis, target marketing, and customer segmentatiion. SuperMartcan anallyze customer information by moniitoring loyallty schemes by membership level, or locattion to send personallized campaiigns.

4. Time_Table

Attribute	Description
time_id (PK)	Unique identifier for each time period
date	Actual date of the transaction.
day	Day of the month (1–31).
month	Month (e.g., January, February, etc.).
quarter	Quarter of the year (Q1–Q4).
year	Year (e.g., 2025).
weekday	Name of the day (e.g., Monday, Tuesday, etc.).
is_weekend	Indicates whether the day is a weekend (Yes/No).

Role:

The Time dimension tablle suports the examiination of salles information based on diferent time levells such as daiily, monthly, quarterlly, or anually. It allows for the time-rellated study of salles, the seasonall trend, and the sales patern analysiss based on diferent days of the week or by quartter.

5. Supplier_Table

Attribute	Description
supplier_id (PK)	Unique identifier for each supplier.

supplier_name	Name of the supplier.
supplier_address	Address of the supplier.
contact_number	Contact information for the supplier.
email_address	Supplier's email address.

2. Design of the Fact Table

The fact table contaiins transactional-levell data that has measureble facts about salles. It is the fact table in the star schema and joiins with dimension tables thorugh foreign keys. It suports anallysis of sales performance by diferent time period, stores, productts, and customers.

Fact Table Name: Sales_Transactions_Table

Attributes of the Fact Table:

Attribute	**Description**
transaction_id	**Primary key. Unique identifier for each sales transaction.**
time_id	**Foreign key linked to the Time dimension. Represents the transaction date.**
store_id	**Foreign key linked to the Store dimension. Indicates where the sale happened.**
product_id	**Foreign key linked to the Product dimension. Identifies the sold product.**
customer_id	**Foreign key linked to the Customer dimension. Identifies who made the purchase.**
quantity_sold	**Numeric fact. Total units sold in the transaction.**
total_sales_amount	**Numeric fact. Total amount generated from the sale (quantity × unit price).**

Foreign Key Relationships:

- store_id → Store dimension
- product_id → Product dimension
- customer_id → Customer dimension

- time_id → Time dimension
-

Note: The Supplier dimension is not directly linked to the fact table but is accessible through the Product dimension, since each product has a supplier_id

Type of Facts and Their Significance:

Fact Column	**Type**	**Explanation**
quantity_sold	Additive	Can be summed across any dimension (e.g., total quantity sold per month).
total_sales_amount	Additive	Can be summed across all dimensions to analyze revenue (e.g., by store or region).

Additive facts are used because they suport **aggregation** over any combiination of dimensions, makeing them esential for busines reporting and anallysis.

3. Aggregate Tables

To enhance querry performance, two agregate tablles may be establiished to precompute frequnet busines measeres

1. Monthly_Sales_Aggregate_Table

Purpose:
Rols up totall sales and quantiity solld for each store and produect by month . this alows for prompt a respond to to such questiions as:

"What are the total sales for each store this month?"

Attributes:

- store_id (FK)
- product_id (FK)
- month
- year
- total_sales_amount (sum of sales)
- total_quantity_sold (sum of quantity)

Relation:

- store_id and product_id link to the Store_Table and Product_Table.
- month and year link to the Time_Table.
- Speeds up monthly sales reporting by precomputing totals.

2. **Yearly_Sales_Aggregate_Table**

Purpose:
Precomputtes totall sales and quantiity sold for each store and productt pre year. This hellps answer questions like:
- “Which store had the highest sales in 2025?”

Attributes:
- store_id (FK)
- product_id (FK)
- year
- total_sales_amount (sum of sales)
- total_quantity_sold (sum of quantity)

Relation:
- store_id and product_id link to Store_Table and Product_Table.
- year links to the Time_Table.
- Speeds up yearly sales reporting by storing precomputed data.

Benefits:
- Improved Performance: Reduces the need to re-calculate totals from the large fact table.
- Faster Reporting: Allows quiick access to summariized sales data by month and year.
- Reduced Load: Minimizes database stran by storing agregate data in precomputed tables.

4. Business Questions

1. Which store had the highest sales last month?
Uses ***Monthly_Sales_Aggregate_Table*** to compare ***total_sales_amount*** by ***store_id.***

This question identifiies the top-performing store by comparing totall monthely sales. It uses pre-agregated data to give instant insiight into store performance.

2. What are the top-selling products by quantity in each region this year?
Uses ***Yearly_Sales_Aggregate_Table*** + joins with ***Store_Table*** to group by region and product.

This question identifiies the top-seling products in each region by looking at yearly salles quantity. It combiines yearly data with stoer information for regionall grouping.

3. Which product categories are generating the most revenue over the year?
Uses Y***early_Sales_Aggregate***_Table + join with ***Product_Table*** to group by category and sum revenue.

This question identifiies which product categores bring in the most revenue by grouping by categry and agregating total salles over the year.

5. Diagram:

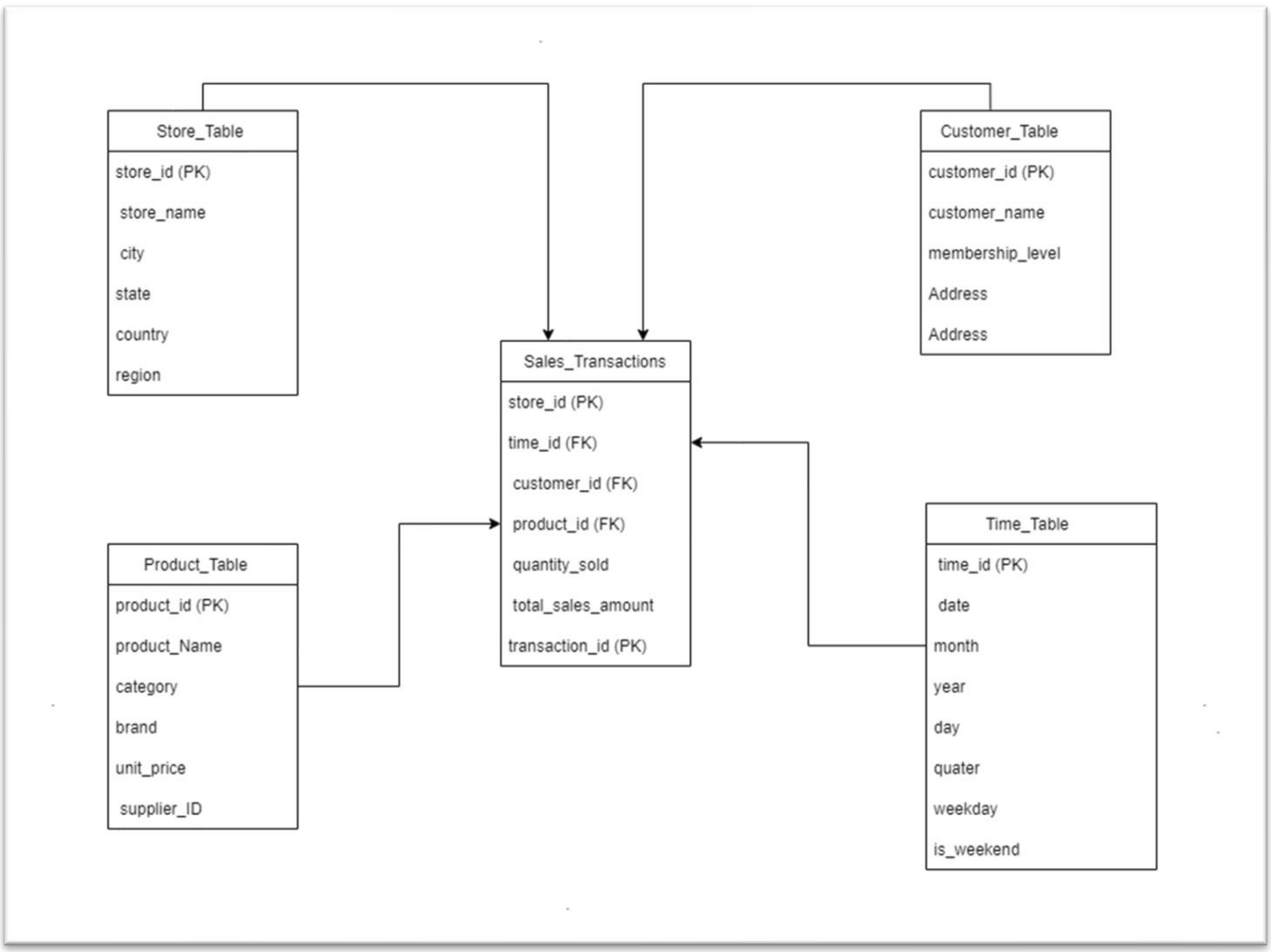

Figure 19: Star scheme

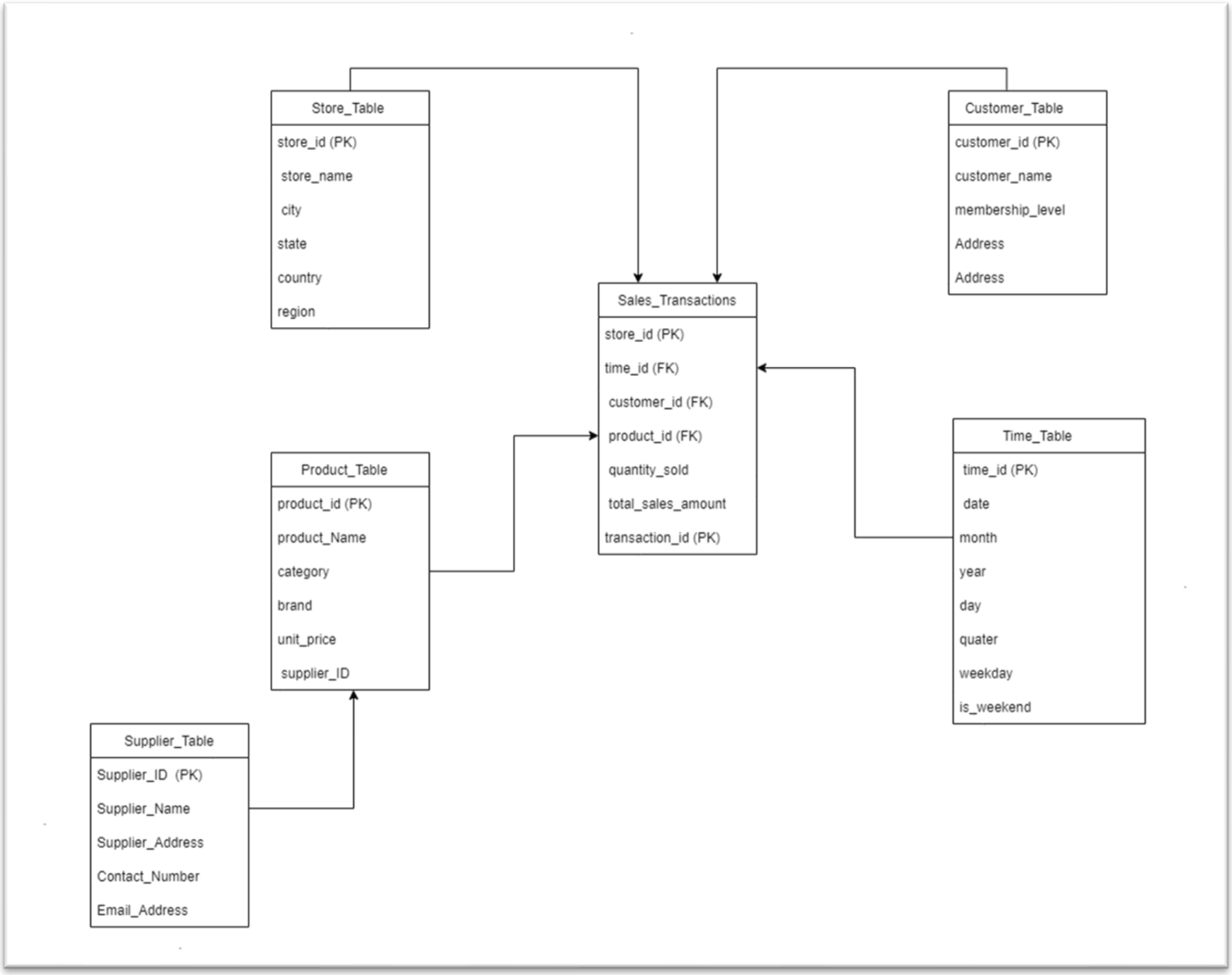

Figure 20:Snowflake Schema

Note on Schema Diagrams:

We have inclluded **two different schema diagrams** for clariity and completenes:

1. **Star Schema** – This is the simplifiied verssion where the **Suppllier Table is excluded**, as it is not directlly connected to the Fact Tablle.

2. **Snowflake Schema** – This versiion inclludes the **Supplier Table**, conected through the Product Table. This normallization reflects rael-world rellationships more acurately.

While ading the Supllier as a separate dimensiion is **optional** in this case, it's a **good design practiice** as it improve data organization and alows suppllier-level analysis when nedded.

Question 3 - Security and Access Framework for HealthAnalytics Inc.

1. Introduction

Protected Heallth Information (PHI) and **Personaly Identiifiable Infromation (PII)** are among the sensiitive health care data that Heallth Analytiics Inc. is curently moving to a clloud- based data lakehouse. In order to suport medicall research and large-scalle analytiics, which wil spur inovation and enhance patiient outcomes, this migartion is essential. However, because health care data is sensiitive, it is esential to make sure that all data is protected in acordance with strict laws like the Generall Data Protection Regullation (GDPR) and the Health Insurance Portabiility and Accountabillity Act (HIPAA).

The scalablle, secure architecture descriibed in this report uses Amazon WebServices (AWS) to protect sensitiive data during the miigration process. In adition to improving data securiity, the architecture wil guarantee that authoried users can efectively acess and analyze data for research and clinical decision-making, while mini mizing the risk of unauthoriized access or data breaches.

2. Authentication vs. Authorization

2.1 Authentication: Multi-Factor Authentication (MFA)

Multi-Factor Authentication (MFA) will be used to guarantee safe acess to the cloud-based data lakehouse. Before being alowed access, MFA requiires users to present two or more forms of veriification, including:

- Something you are familiar with (PIN or password).
- Something you own (OTP sent by email or mobile).

By lowering the posibility of unwanted acess, this technique greatlly improwes the security posture, particullarly for users handlling private medical information. In acordance with security best practiices and complliance requiirements **(HIPAA and GDPR)**, MFA will be enforced using both Azure Active Directory (AD) and AWS Identity and Acess Management (IAM).

2.2 Authorization: Attribute-Based Access Control (ABAC) and Role-Based Access Control (RBAC)

Role-Based Access Control (RBAC) and **Attribute-Based Access Control (ABAC)** will be used in tandem for granullar authorization.

Role-Based Access Control (RBAC): RBAC is in line with the HIPAA "**minimum necessary**" rule and the "**least privilege**" principlle by limitting acces to resources based on the user's assigned rele.

Attribute-Based Access Control (ABAC): ABAC gives you more precise controll over resource acces by enforcing pollicies dynamicaly based on user atributes (like department and shift timiing).

Only authoriized personel can acess sensitiive patient data thanks to these modells, which grant specifiic accass based on rolles and contextual facttors.

2.3 RBAC Matrix

The following table outlines the access permissions for diferent roles within the system:

Role	Permission
Doctors	Full access to assigned patient records, including editing medical history.
Data Scientists	Access to anonymized, aggregated data for analysis, no direct access to identifiable patient data.
Auditors	Read-only access to logs, non-PII fields, and compliance reports.

ABAC Example

- A **Data Scientist** can access data related only to their asigned research demain (e.g., oncology).
- A **Doctor** can view patiient records only during their shifft.

Tols like Apache Ranger and AWS Lake Formatiion, which interfece with IAM to guarante that the apropriate level of acess is aplied based on the user's role and atributes, will be used to enfroce these pollicies. For aded security, sensitiive fields like **Social Security Numbers (SSN)** or Insurance IDs will be hiden from unauthorized users using Dynamiic Data Masking.

3. Data Encryption

To ensure the securiity of sensitive heallthcare data, both at rest and in transit, the folowing encryption methods are recomended:

3.1 Data Encryption at Rest: AES-256

The stanbard for encrypting sensiitive data is AES-256 (Advanc Encryption Standard with a 256-bit key), which makes sure that the confiidentiality of stored data is not jeopardiized by unauthorized acess.

Reasoning:

- AES-256 offers robust security and compllies with GDPR and HIPAA regulations.
- Cloud service providers like AWS and Azure support it extensivey.
- Encryption keys can be managed by cloud services like Azure Key Vaullt or AWS KMS.

3.2 Data Encryption in Transit: TLS 1.3

The most recent version, TLS 1.3 (Transport Layer Security), encrypts data while it is in transit to enable secure network comunication.l

Reasoning:

- Interception and man-in-the-midle atacks are prevented by TLS 1.3.
- Forward secrecy is suported, protecting previous comunications even in the event that future keys are stollen.
- ensures that data transmission complies with GDPR and HIPAA.

Health Anallytics Inc. will protect sensittive healthcare data while adherring to regulatory requirements by deplaying TLS 1.3 for data in transit and AES-256 for data at rest.

4. Access Control

The folowing access controll procedures will be used to limit acess to private information, inclluding Social Security Numbers (SSNs):

4.1 Column-Level Security (CLS):

Goal: Make sure that only authoriized users can access sensitive fiellds, such as SSNs.

Execution:

- SSNs for the patients they are asigned will be fuly accesiible to doctors.
- Only anonymiized data with masked SSNs will be availlable to data scientists.
- Only non- sensitive data with masked SSNs will be visiblle to auditors.

AWS Service: To eficiently manage column-level permisions, AWS Lake Formatiion will be used.

4.2 Dynamic Data Masking (DDM):

The goel is to concael private information from users who lack the requiired authorization.

Execution:Unauthorized users will see SSNs in a masked format (such as "XXX-XX-XXXX").

AWS Service: Depending on user rolles, dynamic data masking will be appllied using AWS Redshift.

By limiting acess to sensitiive data to authorized personel, this policy upholds HIPAA and GDPR complliance.

5. Compliance & Auditing

Protecting sensittive healthcare data requiires adherence to legal frameworks like HIPAA and GDPR. HealthAnalyttics Inc. must put in plece relliable audiiting and monitorring systems in adition to data retention guidellines in order to satisffy these requiirements. To guarantee complliance, the folowing actions will be taken.

5.1 Audit Trails

Goal: To guarantee acountabillity and transpaerncy, keep thorough records of all acceses to and changes made to sensitiive healthcare data.

Execution:

- Turn on thorough loging for all queries, updates, and acceses involving private patiient infromation.
- Logs will document user activitties, including who acessed the data, what was accesed, and when.

AWS tool:
All API calls are loged using **AWS CloudTrail**, which creates an unchangeablle audit traiil of user interacttions with the data lakehouse enviornment.

5.2 Data Retention Policies

Goal: In acordance with HIPAA and GDPR, securelly delete sensitiive data ater it is no longer needed and only kep it for as long as is necesary.

Execution:

- Pollicies for data retention will be establlished to automaticaly archive or remove data in acordance with retention requirenents.
- The "Right to be Forgoten" under the GDPR will be put into efect, enablling patients to aek for their data to be delleted whenever they so chose.

AWS Toools:
Using **AWS S3 Lifecycle Policies**, patient data can be automaticaly archived or deleted acording to predettermined retention periads.

5.3 Real-Time Monitoring

Goal: Prevent posible securitty breaches by imediately identiffying and reacting to suspicious actiivity.

Execution:

- Keep an eye out for any atempts at ilegal acces, odd user behavior (such as loging in from odd places), or acces beyond roles that have been asigned.
- Set off alerts in responses to questionablle activity, guaranteing prompt actiion.

AWS Tools:

- **AWS GuardDuty:** For ongoiing security threat moniitoring.
- Using **AWS CloudWatch**, adminiistrators can create custom allerts based on security threshollds to be informed of unusuall activity.
- **SIEM tools (**such as **IBM QRadar and Splunk**): For combining and examiining log data in order to find any iregularities or security incidnets.

HealthAnalytics Inc. will guarante complete compliiance with HIPAA and GDPR regullations by puting these auditiing and complliance procedures into place. This will shielld private medieal information from unwanted acess and guarante that it is only kept for as long as is requirred.

6. Third-Party Integration

The folowing actions willl be taken in order to safelly provide a resaerch partner with agregated non-PII data:

6.1 API Gateway

The AWS API Gateway willl be used to maange acces to non-PII data. This preserves sensiitive data whille guaranteing that only agregated, non-PII data is availlable. Strict security controlls for data exposuer will be enforced by the API Gateway, which willl act as the interfece between the research parttner and the data.

6.2 Token-Based Access

Token-based authentiication will be used to make suer that only people with permision can acess the data. Uniique API tokens creatted using AWS Cogniito or OAuth 2.0 will be given to research parttners, giving them temporarry, secure acces to the data. Only verifiied users are alowed access thanks to this token-baseed system.

6.3 Rate Limiting

Rate limitting will be used to guard agaiinst abues of the API and guarantee equitablle use. For instance, externall users who acess the data via tha API Gateway will be subject to a liimit of 100 requests per minutes. By doing this, you can keeps the system operatting smothly and avoid overloed or deniial of service atacks.

6.4 Encryption & Secure Communication

TLS 1.3 will be used to secure the data whille it is in transit. Every API interactiion will take plece over HTTPS, guaranteing that all corespondence betwen the data lakehouse and the resaerch parttner is encrrypted and shiellded from posible interception.

By utillizing AWS API Gateway, token-based access, rate limitting, and encryption, this aproach guarantes that the research parttner can safelly access the combiined, non-PII data without jeopardiizing sensitiive data.

7. Architecture flowchart

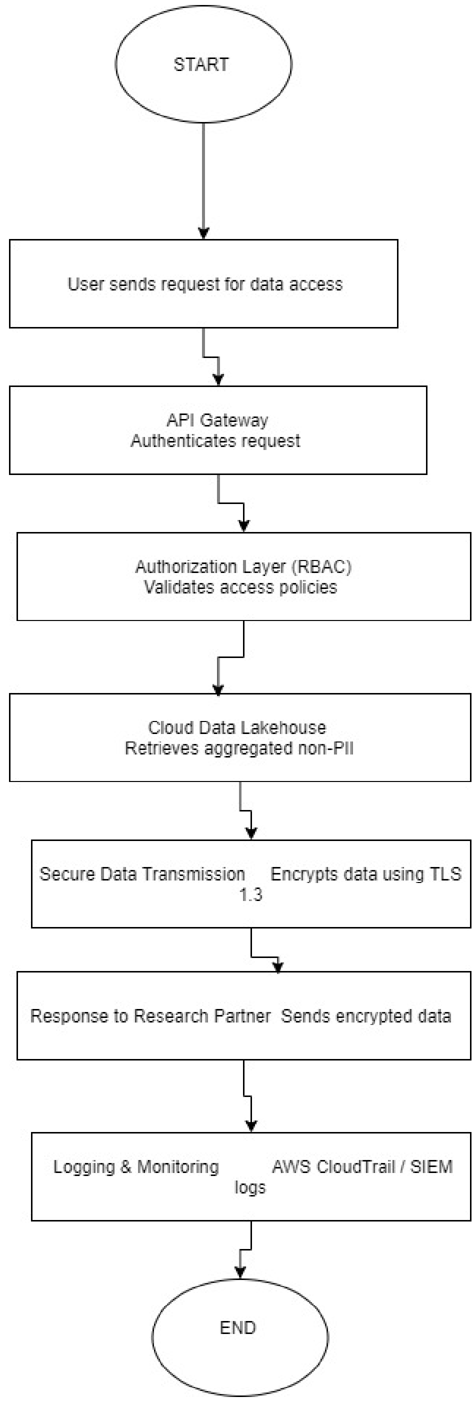

Figure 21: Architecture flowchart

8. Conclusion

In concllusion, HealthAnalytics Inc. must move sensiittive mediical datta to a clloud-based datta lakehousee in oder to suppport research and enhence patiant outcomes. To ensure that patiant data is shiellded from abuse, unauthariized accass , and potantial braeches , it is imperattive that this migretion be carriied out in compllete compliance with legal requiirements such as **HIPAA and GDPR**.

HealthAnalytics Inc. can guarentee that only authoriized personnal have access to sensitiive data by putting in place a strong securitty and accass framework that inclludes column-llevel securitty, dynamicc data maskiing, encryption techniques like **AES-256** and **TLS 1.3** , Multi-Factor Authantication (**MFA**) , Role-Bassed Accass Control (**RBAC**), and Attribute-Bassed Accass Control (**ABAC**) . These actions enhence the organiizetion's overal securiity posture in additiion to meeeting compliance requirements .

Audiit trails, data retantion guidelins , and real-time monitoriing with **AWS CloudTrail, AWS GuardDuty, and SIEM** tools will all help to securre the data and guarntee that any quastionable actiivity is promptly identiified and addressed . Lastly , HealthAnalytics Inc . can securely share aggregated, non-PII data for research purposes without jeopardiizing the integrity of sensitive information by utiliizing a secure **API gateway** , token-based authentication , and rate limiting for third - party integrations .

In the end, this all-encompasing securitty and access framework will allow HealthAnalytics Inc . to function effactively in a safe, legal mannar while cultiveting an atmosphare that encoureges creativity in medical research.

References

- AWS, n.d. ***AWS-EC2 Instances.*** [Online]
  Available at: https://eu-north-1.console.aws.amazon.com/ec2/home?region=eu-north-1#Instances:

- dags, n.d. [Online]
  Available at: https://airflow.apache.org/docs/apache-airflow/stable/core-concepts/dags.html

- Documentation, A. A., n.d. *Apache Airflow Documentation.* [Online]
  Available at: https://airflow.apache.org/docs/

- OpenWeather, n.d. *OpenWeather API.*. [Online]
  Available at: https://openweathermap.org/current

- Services., A. W., n.d. *mazon Simple Storage Service Documentation.*. [Online]
  Available at: https://docs.aws.amazon.com/s3/

- Services, A. W., n.d. *Getting started with Amazon S3.*. [Online]
  Available at: https://aws.amazon.com/getting-started/hands-on/

- https://www.youtube.com/watch?v=uhQ54Dgp6To
- https://youtu.be/5peQThvQmQk
- Draw.io for the digrams -- https://app.diagrams.net/
- Current weather data -- https://openweathermap.org/current
- AWS-EC2 Instances. -- Instances | EC2 | eu-north-1
- ETL Process in Data Warehouse | GeeksforGeeks
- Directed Acyclic Graph in Compiler Design (with examples) | GeeksforGeeks
- S3 buckets | S3 | eu-north-1
- *Apache Airflow Documentation.*--https://airflow.apache.org/docs/apache-airflow/stable/core-concepts/dags.html
- AWS Command Line Interface Documentation
- The Security Rule | HHS.gov
- General Data Protection Regulation (GDPR) – Legal Text
- Data protection in AWS Identity and Access Management - AWS Identity and Access Management
- Data Protection in Lake Formation - AWS Lake Formation
- Security design principles - Security Overview of Amazon API Gateway
- Data Protection and Privacy | AWS

- Guide to TLS Standards Compliance - SSL.com
- Compliance validation for AWS CloudTrail - AWS CloudTrail
- GDPR - Amazon Web Services (AWS)
- Controlling and Managing Access to a REST API in Amazon API Gateway
- How to Setup AWS Multi-Factor Authentication (MFA) and AWS Budget Alerts | Amazon Web Services
- Using Amazon GuardDuty to continuously monitor and secure your AWS AI workloads
- Leacture notes and tutorials
- ChatGPT for Error Handling in Apache Airflow
- ChatGPT for Enhancing English Writing
- airflow.hooks.S3_hook — Airflow Documentation
- python - How to use the s3 hook in airflow - Stack Overflow
- FileSensor — Airflow Documentation
- What are the design schemas of data modelling? | GeeksforGeeks