Part 2: Conceptual Write-Up

Prepare a brief technical note (400–500 words max) addressing the following:

What is Apache Airflow, and how does it work?

Apache Airflow is an open-source workflow orchestration platform designed for programmatically authoring, scheduling, and monitoring workflows. It represents workflows as Directed Acyclic Graphs (DAGs), where nodes are tasks and edges define dependencies. Workflows are defined in Python code, making them dynamic and reusable. The Airflow scheduler decides when tasks should run, while executors (e.g., Celery, Kubernetes) handle task execution. Logs and status updates are viewable in the Airflow web UI, providing transparency.

Where does Airflow fit in modern data engineering workflows?

In modern data engineering, data pipelines need to automate ingestion, transformation, and loading across various systems. Airflow fits in as the central orchestrator: it schedules batch ETL jobs, manages dependencies, and coordinates execution across distributed environments. For example, it can run a daily job to extract sales data from APIs, transform it with Spark, and load it into a warehouse like Snowflake. Airflow also integrates well with cloud platforms (AWS, Azure, GCP), making it a key part of production-grade data platforms.

How is Airflow different from traditional schedulers or other tools like Prefect or Luigi?

Unlike traditional cron-based schedulers, Airflow is workflow-aware—it understands dependencies, retries, and task states. Prefect emphasizes simplicity and cloud-native orchestration, while Luigi focuses on dependency management for Python tasks. Airflow stands out for its flexibility, large ecosystem of operators, and scalability in enterprise contexts. It provides a central UI, monitoring, and integrations, whereas cron or simple schedulers lack these advanced features.

What are the key components (e.g., DAGs, operators, scheduler, executor) and how do they interact?

- **DAGs:** Python-defined workflows representing tasks and dependencies.
- **Operators:** Predefined templates (PythonOperator, BashOperator, SQL operators, etc.) for executing specific tasks.
- **Scheduler**: Determines when tasks should run based on schedules and DAG definitions.

- **Executor**: Executes tasks across resources (Local, Celery, Kubernetes, etc.).
- **Webserver (UI)**: Provides a monitoring dashboard for DAG runs, logs, and task states. Together, these components ensure that workflows are executed in order, retried on failure, and monitored for transparency.

Based on your learning, where do you see Airflow being useful in real-time enterprise or product scenarios?

Airflow is widely used in ETL Pipelines: Automating ingestion from APIs/databases, transformation with Spark/Pandas, and loading into warehouses. Machine Learning: Orchestrating model training, validation, and deployment pipelines. Data Warehousing: Scheduling batch loads into Snowflake, Big Query, or Redshift. Business Reporting: Automating daily/weekly report generation and email delivery. IoT & Logs Processing: Coordinating large-scale data ingestion from sensors or logs into analytics platforms.