Problem Statement:-

The goal is to create a Blockchain Analytics Data Warehouse that can centralize, structure, and process blockchain data for cryptocurrencies to support various stakeholders in making informed decisions, improving operational efficiency, and ensuring compliance.

Basics of BlockChain:-

Data Engineering Datawarehouse Design

Steps to design a datawarehouse

1. Understand the business usecase:- Identify the business processes that the warehouse will support and with stakeholders gather requirements for each process.
2. Identify the grain:- For each process define the grain at which data is captured and make sure it is consistent across the data warehouse
3. Design the fact table:-
4. Design the dimension table:-
5. Develop ETL Processes:- That is Extract the data from origin, transform it so it fits in data warehouse and load it in data warehouse. You can do ETL or ELT. After ELT we also have to do data cleaning
6. Identify Slowly Changing Dimensions:- Plan for changes in dimensions. SCD 1 is update, SCD 2 is maintain all the new records along with older ones like versionong, SCD 3 is where we can get new additional information
7. Testing and Validation:- Test the data warehouse so that it reflects the true source data and validate it to check if the queries run agaist the warehouse gives expected results.
8. Documentation:- Document the design ie the schema, ETL processes, and business rules used.

We have no duplicates in dimension tables. When assigning the number of characters for character data types unnecessary giving a lot of characters will consume a lot of memory.

RedShift Cluster:-

Amazon Redshift is a fully managed, petabyte-scale data warehouse service on AWS. It is designed for **large-scale data storage, querying, and analytics**. A **Redshift cluster** is a collection of compute nodes organized to store and process data in a parallel and distributed manner. It will use sql, utilize columnar storage, and parallel execution. We need to do configurations as which type of nodes we need and the number of nodes we need. Redshift cluster is always created on a subnet inside the vpc. Of all the nodes there will be a leader node which performs query planning, optimization, and coordination across compute nodes. We apply security groups to the cluster. Redshift nodes use local storage (e.g., SSDs) or managed storage (RA3 nodes) for storing the data.  AWS initializes the **PostgreSQL-compatible database engine** on the leader node. A default database (e.g., dev) is created, and metadata about tables, schemas, and users is stored.

Redshift is optimized for **OLAP (Online Analytical Processing)** workloads. To achieve this:

* Queries are distributed by the leader node to compute nodes in parallel.
* Data is stored in slices (subsections of nodes) to allow efficient processing.

**How Redshift Executes Queries in a Cluster**

1. **Query Submission**:
   * A user submits a query to the leader node (via tools like the Redshift Query Editor or a client like SQL Workbench).
2. **Query Parsing and Optimization**:
   * The leader node parses the query, generates a query plan, and determines the most efficient way to retrieve the requested data.
3. **Parallel Execution**:
   * The leader node distributes the query into smaller tasks and assigns them to compute nodes.
   * Compute nodes process the tasks in parallel using the data stored in their local storage.
4. **Result Aggregation**:
   * The compute nodes return results to the leader node, which aggregates the results and delivers them to the user.

We create cluster subnet group and add the vpc subnets to it.

Steps taken:-

When starting the project We always start with creating a repository, Use git clone to clone it to your local pc, do git add so as to take snapshot of the files which have changes, then git commit that means the changes are fine and can be pushed to the github repository. Git status gives the untracked files, files which have changes. Always create branches so that the main branch is not affected.

*  When you create a virtual environment using venv or virtualenv, it doesn’t copy all the Python files from the system installation.
* Instead, it **reuses the existing Python installation** by creating symbolic links (or similar references) to the system's Python interpreter and libraries. This is what makes it "lightweight."

 **Package and Dependency Isolation**:

* While the virtual environment uses the **same base Python interpreter**, any **packages you install (via pip) are isolated to the virtual environment**.
* This means packages and versions installed in the virtual environment **do not affect** the global Python environment or other virtual environments.

**venv (Standard Library Module)**

* **Introduced in Python 3.3**, venv is a built-in module used to create lightweight virtual environments.
* Does not require installation since it’s included in Python’s standard library.

**Key Features:**

* **Lightweight**:
  + It’s simple and has fewer features compared to virtualenv.
* **Cross-platform**:
  + Works consistently across Windows, macOS, and Linux systems.
* **Default Tool**:
  + If you’re using Python 3.3 or later, venv is the default choice for creating virtual environments.

**Limitations:**

* **Backward Compatibility**:
  + Only works with Python 3.3 and later.
* **No Extra Features**:
  + Doesn’t support advanced features like the ability to relocate virtual environments or extended shell customization.

**virtualenv (Third-Party Package)**

* **Older than venv**, virtualenv is a third-party tool that predates Python 3.3 and is designed to work with Python 2 and Python 3.

**Key Features:**

* **Backward Compatibility**:
  + Works with older versions of Python, including Python 2.x.
* **Rich Features**:
  + Offers more advanced options, such as relocating virtual environments or specifying a Python version.
* **Customizable**:
  + Provides more flexibility with environment creation and setup.
* **Active Development**:
  + Frequently updated with new features and fixes.

**Limitations:**

* **Requires Installation**:
  + Not included by default in Python; must be installed with pip.

After creating a venv environment we need to install ipykernel and Next, register your virtual environment as a new kernel in Jupyter: