Azure Health Care Data Engineering Project

Project Overview

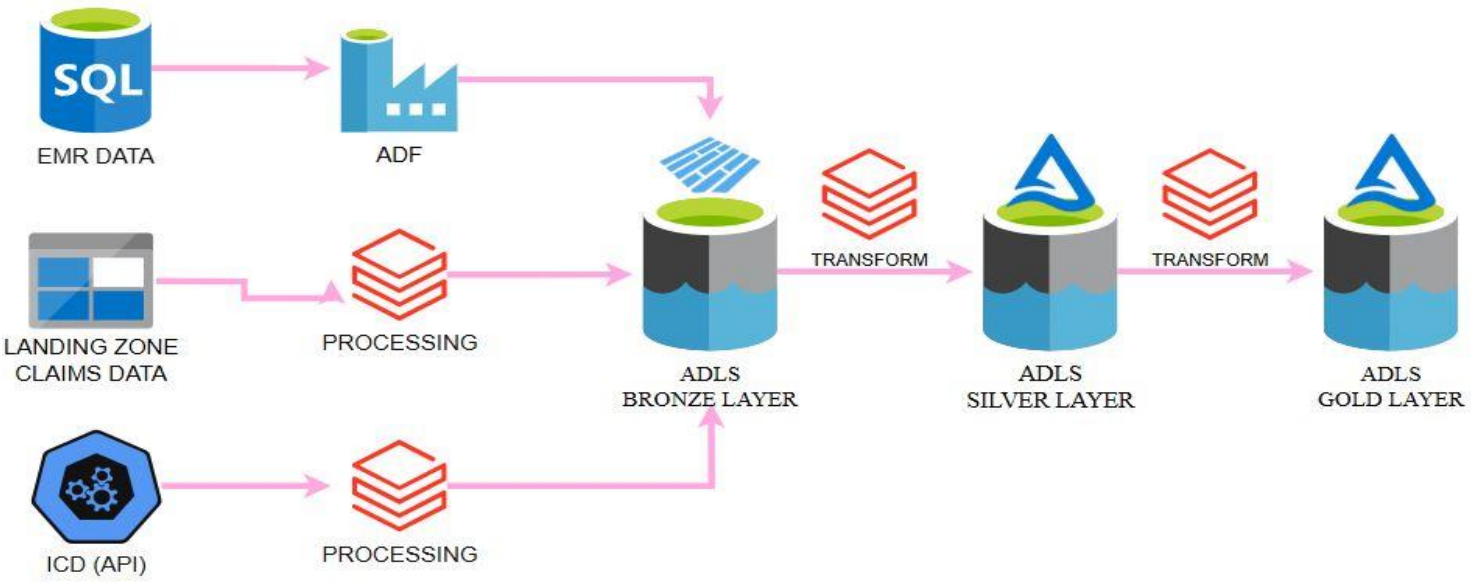
Revenue Cycle Management (RCM): The process hospitals use to manage financial aspects, from patient appointment to provider payment. It involves accounts receivable (money owed to the hospital) and accounts payable (money the hospital owes). The project focuses more on accounts receivable to ensure financial health.

- **It uses a medallion architecture.**
- **The goal is to build a data pipeline that transforms raw data into fact and dimension tables for reporting and analysis.**
- **Metadata-driven approach allowing the pipeline to dynamically adapt to changes in the data environment without requiring code modifications. By centralizing configurations in a metadata file, the pipeline is more robust, maintainable, and scalable.**
- **Common data model (CDM) to handle variations in schemas from multiple data sources, primarily focusing on integrating data from different hospitals that may have different table structures or naming conventions**

The project utilizes various Azure services including

1. **Azure Data Factory (ADF)**
2. **Azure Databricks**
3. **Azure SQL DB**
4. **Azure Data Lake Storage Gen2 (ADLS Gen2)**
5. **Key Vault**

Pipeline Diagram



Key Concepts

- **Medallion Architecture: A data architecture pattern with distinct layers:**
  - **Bronze: Raw data as it comes from the source.**
  - **Silver: Cleaned, conformed and enriched data.**
  - **Gold: Fact and dimension tables for reporting.**

- **Fact and Dimension Tables:**
  - **Fact tables contain numeric values that represent a business event (e.g. transaction amount).**
  - **Dimension tables contain descriptive attributes related to the facts (e.g., patient, provider, department).**
- **Slowly Changing Dimension (SCD) Type 2: A method to track historical changes in dimension data, keeping a record of all changes over time.**

Data Sources in Detail

- **Electronic Medical Records (EMR) Data:**
  - Originates from two separate Azure SQL Databases representing two hospitals.
  - Includes tables like **patients, providers, departments, transactions, and encounters**.
  - These tables are extracted to the bronze layer.
- **Claims Data**:
  - Sent from insurance companies, it arrives as flat files (CSV) and is stored in the **landing zone** before being moved to the bronze layer.
  - Contains information about **patient claims, payments, and insurance details**.
- **NPI (National Provider Identifier) Data:**
  - Obtained from a public API and includes information about **doctors and healthcare providers**, like their unique identifiers and specializations.
  - The data is directly pulled into the bronze layer.
- **ICD (International Classification of Diseases) Codes:**
  - Pulled from a public API; provides a standardized system for classifying diseases and diagnosis codes.
  - These codes are also directly loaded into the bronze layer.
- **CPT (Current Procedural Terminology) Codes**:
  - Flat files (CSV) from a third-party vendor, stored in the landing zone, and then processed into the bronze layer.
  - CPT codes describe the procedures performed by healthcare providers.

Key Technologies & Tools

- **Azure Data Factory (ADF):**
  - Used to orchestrate the data pipeline and call Databricks notebooks.
  - Uses link services, data sets, and activities to ingest and transform data.
  - The pipeline uses a lookup activity to read metadata from the config file.
  - A "for each" loop iterates through tables specified in the config file.
- **Azure Databricks:**
  - Used for data processing, cleaning, transformation, and implementing SCD type 2.
  - Notebooks are used to execute data transformations and load the data to each layer.
- **Azure SQL DB:**
  - Stores the EMR data.
- **Azure Data Lake Storage Gen2 (ADLS Gen2):**
  - Stores data in landing, bronze, silver, and gold layers.

- **Azure Key Vault:**
  - Used for storing connection strings and passwords securely.
- **Unity Catalog:**
  - **Unity Catalog** is used as a central metastore to manage metadata across different workspaces. This helps to maintain a centralized repository of tables, and schemas, and provides data lineage tracking capabilities.
  - Lineage is enabled to track the flow of data between tables and layers.

Setting up the Foundation

- **Domain Understanding:**
  - RCM involves managing hospital revenue, including patient appointments and provider payments.
  - Key metrics or **KPIs** include **days in accounts receivable (AR)** and **net collection rate**, that help assess the effectiveness of the RCM process.
- **Setting up Azure Resources:**
  - An **Azure Data Lake Storage Gen2 (ADLS Gen2)** account is created. The hierarchical namespace is enabled, turning it into an ADLS Gen2 account for better analytics workloads.
  - Containers are created within the storage account for different stages of the data pipeline including **landing, bronze, silver, and gold**, plus an additional container named **configs** for configuration files.
- **Creating a Metadata-Driven Pipeline:**
  - A **configuration file** (loadconfig.csv) is created to make the pipeline generic and metadata-driven. The file is stored in the configs container and includes information on table names, source, target paths, load types, and watermark columns.
  - The configuration file helps the system to know what data to extract, where to store, and how to process it.
- **Configuring Azure Data Factory (ADF):**
  - **Linked Services** are set up in ADF to connect to various data sources and destinations.
    - This includes connections to the Azure SQL DB (source), the ADLS Gen2 (target), and Data Bricks Delta Lake.
    - **Connection parameters**, like the server's name, authentication type, username and passwords, are configured to allow data access.
  - **Datasets** are configured, defining the structure and schema of data for various sources and destinations.
    - Datasets are created for Azure SQL DB, flat files, Parquet format, and Azure Data Bricks Delta Lake.
    - These datasets are generic, with parameters for file paths, container names, table names, and schema names, to be supplied at runtime.

    - An **audit table** is created using Azure Data Bricks Delta Lake, to track the execution status of the pipeline.
- **Implementing the Data Ingestion Pipeline:**
    - The pipeline starts with a **Lookup activity** to read the config file from the ADLS Gen2 configs container.
    - A **For Each activity** is used to iterate through each entry or row in the config file. The metadata from each row is extracted and used by subsequent activities.
    - A **Get Metadata activity** checks if a corresponding parquet file already exists in the bronze layer for a given table. If the file exists, it's moved to an archive folder.
    - The pipeline uses an **If condition** to decide between full load and incremental load based on the config file.
        - For **full load**, data is extracted from the SQL database and written to ADLS Gen2 in Parquet format.
        - For **incremental load**, the pipeline uses a **lookup activity** to fetch the last modified date from the audit table, and then only fetches new data.
    - Finally, the pipeline updates the audit table with information about each run
- **Extending Data Sources:**
    - **Claims data** from insurance providers is moved from the landing zone to the bronze layer. An extra column for **data source** is added using the file name.
    - **CPT codes** are moved from landing to bronze, while renaming the columns to lower case and using underscore for spaces.
    - **NPI and ICD codes** are obtained from public APIs and written directly to the bronze layer using a data bricks notebook, with appropriate schema applied.
- **Implementing Azure Key Vault:**
    - **Azure Key Vault** is created and configured to store credentials securely.
        - Secrets are created in the Key Vault for SQL DB, ADLS Gen2 and Data Bricks.
        - Access policies are set up in Key Vault to authorize services like Data Factory and Data Bricks to access the secrets.
    - **ADF Linked Services** are updated to retrieve credentials from the Key Vault instead of hardcoding them.
    - A **scope** is created in Databricks using the key vault resource id to authorize access to key vault secrets.
- **Improving Pipeline Efficiency and Best Practices:**
    - The pipeline is modified to run in **parallel** by removing the auto-increment key from the audit table, allowing multiple tables to be processed simultaneously.
    - An **is active flag** is implemented, to control which tables are processed.
    - **Retries** are implemented to handle transient errors or network issues in ADF.
    - **Mount points** are created using Databricks file system for accessing storage locations programmatically, by retrieving the access keys from key vault.
- **Implementing the Silver Layer:**
    - Data is read in Parquet format from the bronze layer, through Databricks notebooks.

    - **Data quality checks** are performed to identify and flag bad data by using an is_quarantine flag.
    - A **common data model (CDM)** is implemented to unify schema differences across different hospitals. This includes column renaming, and creating surrogate keys where necessary.
    - **SCD type 2** is implemented, to maintain historical data changes in tables like patients, transactions, claims, and encounters.
    - Data is stored in **Delta tables**, which improve data management and provide ACID transactions.
- **Implementing the Gold Layer:**
    - Data is read from the silver layer in Delta format.
    - Data is filtered by the is_current flag, to get latest records, and by the is_quarantine flag to avoid bad records.
    - **Fact and dimension tables** are created based on the requirements of reporting and analytics. The fact table (transaction) stores numeric and quantitative data, while the dimension tables (patient, provider, department) store descriptive information.
- **End-to-End Pipeline Orchestration:**
    - A single **end-to-end pipeline** in ADF is created to integrate data ingestion, transformation, and loading processes.
        - The end-to-end pipeline executes EMR source to bronze pipeline first and then invokes Azure Databricks notebooks to process data through silver and gold layers.
    - The pipeline runs in parallel to expedite the processing.