**🧠 Phase 2: Data Architecture Design**

In this phase, we'll create the blueprint for your data warehouse. We will define the overall structure, plan the data flow, and design the schemas for our different layers. The goal is to create a design that is scalable, efficient, and directly supports the business challenges you've outlined.

Here are the key tasks for this phase:

1. **Choose a High-Level Architecture:** We'll confirm the use of the Medallion Architecture (Bronze, Silver, Gold layers) within SQL Server.
   * **🥉 Bronze Layer (Raw Ingestion):** This layer will hold the raw, unaltered CSV data. Its primary purpose is to be a faithful copy of the source system data. This provides a historical archive and allows us to rebuild downstream layers without re-querying the source ERP system.
   * **🥈 Silver Layer (Cleansed & Conformed):** Here, we will clean the raw data, apply business rules, handle missing values, and standardize data types. The data will be modeled into well-defined tables (e.g., DimPatient, DimDoctor, FactAppointment).
   * **🥇 Gold Layer (Aggregated & Business-Ready):** This layer contains data that is aggregated and optimized for reporting. We'll create tables or views specifically designed to answer the business questions you listed (e.g., MonthlyPatientVisits, AppointmentWaitTimeAnalysis). This is the layer Power BI will connect to.
2. **Design the Database Schema:**
   * **For the Bronze Layer:** We'll start by designing a staging table in your SQL Server database that exactly matches the columns in your CSV file. We'll add some metadata columns like LoadDate and SourceFileName for auditability.
   * **For the Silver & Gold Layers:** We will begin designing a dimensional model (Star Schema), which is ideal for analytics. This involves identifying your **Facts** (the numerical measures like wait times, costs) and **Dimensions** (the contextual attributes like patient details, dates, departments).
3. **Document the Design:** We will create data flow diagrams and document the schema designs. This documentation is vital for maintainability and for onboarding new team members.

**Sampling the Outpatient Data**

I'm now sampling the data from the CSV file, outpatient\_data\_realistic.csv. My focus is on quickly identifying column names and data types. This initial sampling will drive the Bronze layer design, ensuring a foundational structure to store the raw data effectively. I'll need to recognize any nuances soon.

**Defining the Bronze Schema**

I've transitioned to using pandas to read and sample the outpatient\_data\_realistic.csv file. I plan to use .head() to inspect the initial rows. This information will inform a preliminary CREATE TABLE statement for the Bronze layer. It's a quick and practical approach.

**Reviewing Data Structure**

I've just successfully loaded the CSV file. I've taken a quick glance at the column names, and I can confirm that they include: AppointmentID, AppointmentDateTime, PatientID, NHS\_ID, PatientFirstName, PatientLastName, and PatientDOB. Now I will begin to analyze the data within each column.

**Detailing the Structure**

Having loaded the data, I can confirm the column names: AppointmentID, AppointmentDateTime, PatientID, NHS\_ID, PatientFirstName, PatientLastName, PatientDOB, PatientPostcode, Clinic Name, Specialty, Provider ID, ProviderName, Diagnosis Code, DiagnosisDescription, and Appointment Status. Most are object datatypes, with NHS\_ID as int64. Date/time columns require future Silver layer transformation. I'm now crafting the T-SQL Bronze layer table.

**Crafting Bronze Table**

I've finalized the Bronze layer table structure. The T-SQL CREATE TABLE script is ready for review. I've chosen NVARCHAR for string columns for flexibility. Metadata columns (LoadDateTime, SourceFileName) are incorporated for auditing and lineage. I'm hoping you'll give the script a look and give me the go ahead!

Python

import pandas as pd

# Load the CSV file into a pandas DataFrame

try:

df = pd.read\_csv('outpatient\_data\_realistic.csv')

print("File loaded successfully. Here's a sample of the data:")

print(df.head())

print("\nHere is the data type information for each column:")

df.info()

except FileNotFoundError:

print("The file 'outpatient\_data\_realistic.csv' was not found.")

except Exception as e:

print(f"An error occurred: {e}")

Code output

File loaded successfully. Here's a sample of the data:

AppointmentID AppointmentDateTime PatientID NHS\_ID PatientFirstName PatientLastName PatientDOB PatientPostcode ClinicName Specialty ProviderID ProviderName DiagnosisCode DiagnosisDescription AppointmentStatus

0 A1001 12/07/2025 07:58 P537 7695140767 Michael Hendrick 22/02/1948 NP20 9WE Royal Glamorgan Gastroenterology D88 Dr. Davies S82.1 Fracture of upper end of tibia Attended

1 A1002 12/07/2025 07:58 P531 3900866457 Eugene Rhodes 11/05/1965 CF30 6NZ Singleton Hospital Oncology D21 Dr. Evans K21.9 Gastro-oesophageal reflux disease DNA

2 A1003 12/07/2025 07:58 P531 3900866457 Eugene Rhodes 11/05/1965 CF30 6NZ Royal Glamorgan Dermatology D67 Dr. Williams L20.9 Atopic dermatitis, unspecified Attended

3 A1004 12/07/2025 07:59 P570 2367876150 Denise Shaw 14/09/1974 NP23 1OE University Hospital Cardiology D12 Dr. Thomas M54.5 Low back pain DNA

4 A1005 12/07/2025 07:59 P628 6755009776 Mitchell Cantrell 24/09/2010 CF32 5RW Royal Glamorgan Dermatology D33 Dr. Roberts I10 Essential hypertension Attended

Here is the data type information for each column:

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 930 entries, 0 to 929

Data columns (total 15 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 AppointmentID 930 non-null object

1 AppointmentDateTime 930 non-null object

2 PatientID 930 non-null object

3 NHS\_ID 930 non-null int64

4 PatientFirstName 930 non-null object

5 PatientLastName 930 non-null object

6 PatientDOB 930 non-null object

7 PatientPostcode 930 non-null object

8 ClinicName 930 non-null object

9 Specialty 930 non-null object

10 ProviderID 930 non-null object

11 ProviderName 930 non-null object

12 DiagnosisCode 930 non-null object

13 DiagnosisDescription 930 non-null object

14 AppointmentStatus 930 non-null object

dtypes: int64(1), object(14)

memory usage: 109.1+ KB

Excellent. We have now sampled the data from your outpatient\_data\_realistic.csv file. This gives us the exact structure we need to design the Bronze layer table.

Here are the columns we've identified from the file:

* AppointmentID
* AppointmentDateTime
* PatientID
* NHS\_ID
* PatientFirstName
* PatientLastName
* PatientDOB
* PatientPostcode
* ClinicName
* Specialty
* ProviderID
* ProviderName
* DiagnosisCode
* DiagnosisDescription
* AppointmentStatus

**🥉 Phase 4: Bronze Layer Development (Initial Step)**

We are now ready to take the first step in **Phase 4** by creating the initial table in our data warehouse.

**Designing the Bronze Staging Table**

The goal for the Bronze layer is to ingest the data exactly as it is, without any transformations. This creates a raw, historical copy. We will create a staging table in a dedicated SQL Server schema, which we can call Bronze.

For flexibility and to prevent loading errors from unexpected data types, we'll define most columns as NVARCHAR(MAX). We can enforce stricter data types later in the Silver layer. We will also add two crucial metadata columns:

* LoadDateTimeUTC: To record when the data was loaded.
* SourceFileName: To track which file the data came from.
* We have our destination table, Bronze.stg\_OutpatientData, ready. Now, we'll write the T-SQL script to perform the actual data loading. This is a core task in our **Bronze Layer Development**.
* **🥉 Phase 4: Bronze Layer Development (Data Ingestion)**
* For this task, we will use the BULK INSERT command in T-SQL. It's a highly efficient way to import data from a flat file directly into a SQL Server table. It's perfect for our goal of getting the raw data into the Bronze layer quickly and with minimal overhead.
* **Stored Procedure for Loading the Bronze Layer**
* SQL
* /\*
* ===============================================================================
* Stored Procedure: Bronze.Load\_Staging\_OutpatientData
* ===============================================================================
* Script Purpose:
* This stored procedure orchestrates the loading of outpatient data into the
* Bronze layer. It performs the following actions:
* 1. Truncates the 'Bronze.stg\_OutpatientData' table to ensure a fresh load.
* 2. Uses the BULK INSERT command to efficiently load data from the source
* 'outpatient\_data\_realistic.csv' file.
* 3. Updates the metadata columns (SourceFileName) for the newly loaded batch.
* 4. Logs the process, including timing and success or error messages.
* Parameters:
* None. This stored procedure is designed to run as a single, complete step.
* Usage Example:
* EXEC Bronze.Load\_Staging\_OutpatientData;