Azure Data Pipeline Documentation

This document details the implementation of an Azure data pipeline designed to populate a SQL database with data from CSV files. The pipeline leverages Azure Data Factory's data flow capabilities for data transformation and loading.

1. Infrastructure Setup

The following Azure resources were provisioned for this project:

- **Resource Group:** A resource group was created to logically group all related Azure services. This simplifies management and deployment. *This name is not shown in the screenshots.*
- **SQL Server:** An Azure SQL Server instance was created to host the target database. *This information is also not available from the screenshots.*
- **SQL Database:** A SQL database named *[database name not visible in screenshots]* was created within the SQL Server instance. This database stores the ingested and transformed data. The following tables were created within the database:
    - **Customers:** Stores customer information including CustomerID and Country.
    - **Products:** Stores product information including StockCode, Description, and UnitPrice.
    - **Invoices:** Stores invoice information including InvoiceNo, CustomerID, and InvoiceDate.
    - **InvoiceItems:** Stores details of items within each invoice including InvoiceNo, StockCode, Quantity, and TotalPrice. This table also has an auto-incrementing ID column as the primary key.
-

2. Data Factory Implementation

An Azure Data Factory instance was created to orchestrate the data ingestion and transformation process. The factory contains the following key components:

- **Linked Services:** Linked services were established to connect to the source data (CSV files) and the destination SQL database. These linked services define the connection strings and authentication details required to access the respective data stores. *These are not visible in the provided screenshots, but they are essential for the Data Factory to function.*
- **Datasets:** Datasets represent the structure and location of the data being processed. Four datasets were created, corresponding to each of the CSV files (Customers, Products, Invoices, InvoiceItems) and implicitly for the destination tables in SQL Database. These datasets likely specify the file format (CSV), delimiters, and column mappings.

- **Data Flows:** Data flows handle the transformation logic for each dataset. Four data flows were created, one for each table:
    - **customers1:** Imports data from the "Customers.csv" file and exports it to the Customers table in the SQL database.
    - **products1:** Imports data from the "Products.csv" file and exports it to the Products table in the SQL database.
    - **invoices1:** Imports data from the "Invoices.csv" file and exports it to the Invoices table in the SQL database.
    - **invoiceitems1:** Imports data from the "InvoiceItems.csv" file and exports it to the InvoiceItems table in the SQL database.
      Each data flow includes a source and a sink transformation. The source defines the data source (dataset), and the sink defines the data destination (SQL table). Additional transformations can be added within the data flow to perform operations like data cleansing, filtering, and aggregation. The screenshots show a basic "Import data" source and "Export data" sink. This is where the connection to the defined source and destination datasets is made.

- **Pipeline:** A pipeline orchestrates the execution of the data flows. The "graduationproject" pipeline is responsible for running each data flow.

3. Data Transformation Details

- **Data Type Conversion:** Ensure data types in the CSV files are correctly mapped to the corresponding data types in the SQL tables. For instance, "UnitPrice" and "TotalPrice" should be converted to decimal, "CustomerID" and "Quantity" to integers, etc.
- **Data Cleansing:** Handle potential issues like null values, invalid characters, and inconsistent data formats.
- **Data Mapping:** Map the columns from the CSV files to the corresponding columns in the SQL tables.

4. Execution Flow

The data pipeline executes the data flows in a specific order, ensuring data integrity through foreign key relationships:

1. **customers1:** Loads data into the Customers table.
2. **products1:** Loads data into the Products table.
3. **invoices1:** Loads data into the Invoices table, referencing the Customers table.
4. **invoiceitems1:** Loads data into the InvoiceItems table, referencing both the Invoices and Products tables.