C# Microsoft SQL Architect Task:

The task is designed to assess your coding prowess, architectural insight, and problem-solving speed. (20 hours)

Please review the task below and if you believe it cannot be completed within the allotted time, do not hesitate to suggest a modified scope. Effective scoping and time estimation is a critical skill for an architect, and we welcome your input. Please do not rely on any boilerplate code.

Your creativity, adherence to best practices, and efficiency will be key evaluation points.

Design and develop a SQL database and C# application that interact with the database, using Azure SQL and Azure Data Factory**.** We can provision your credentials and Azure access. The database should be designed with three tables: Products, Categories, and Orders, with the relationships between the tables defined as follows:

Table 1: Products

* product\_id: unique identifier for the product
* product\_name: name of the product
* category\_id: foreign key to the Categories table
* price: price of the product
* description: description of the product
* image\_url: URL for the product image
* date\_added: date on which the product was added to the website

Table 2: Categories

* category\_id: unique identifier for the category
* category\_name: name of the category

Table 3: Orders

* order\_id: unique identifier for the order
* order\_date: date on which the order was placed
* customer\_name: name of the customer who placed the order

Table Relationships:

* The Products table has a many-to-one relationship with the Categories table, based on the category\_id foreign key.
* The Orders table has a many-to-many relationship with the Products table, based on a join table (not shown in the above schema) that links the two tables.

In addition to the above requirements, the application should implement a search method for products that allows the user to search using all available parameters, including product name, category, price, description, and date added. The search method should use Azure Cognitive Search to provide enhanced search capabilities.

Requirements:

* Use Azure Data Factory (or alternate option if better) to extract data from a source system, transform the data, and load it into the SQL database.
* Use SQL to design and implement a schema for the database, including tables, indexes, and stored procedures.
* Use C# to develop an application that interacts with the database, including implementing the following features:
  + A command-line interface (CLI) that allows the user to initiate and monitor ETL pipelines, search the data using Azure Cognitive Search, and search for products using all available parameters.
  + Code to handle errors and exceptions, and to log application events.
  + Code to optimize SQL queries and database design for performance and scalability.
  + Code to implement CDC to capture specific changes to the data in the database and write them out to another database.
* Use Azure Cognitive Search to provide enhanced search capabilities for the data in the SQL database.
* Implement CDC (or alternate option if better) to capture specific changes to the data in the SQL database and write them out to another database.

There are several tools that can be used to generate large amounts of data for importing into a database. Here are a few options:

1. Microsoft SQL Server Data Generator: This is a tool provided by Microsoft that can be used to generate test data for SQL Server databases. It can generate data in a variety of formats, including CSV, XML, and JSON.
2. Data Factory: Azure Data Factory can be used to generate and populate test data into the database tables. Using ADF, you can define your own data flow pipelines to read data from different sources like CSV, JSON, SQL Server, etc., transform the data, and write it to the destination tables.
3. PowerShell: PowerShell scripts can be used to generate data for importing into the database. You can use scripts to generate test data in a variety of formats, including CSV, XML, and JSON.

Deliverables:

* A GitHub repository containing the SQL database schema, the C# source code, and a README.md file that provides instructions for building and running the application.
* A written report that documents the technical decisions and trade-offs made during the development process, as well as any challenges encountered and solutions implemented.

