MINT | Hiring | SSIS Home Task

Objective:

Develop an SSIS-based ETL process that migrates data from a CSV file to a production SQL Server table, via a staging table. The task aims to demonstrate the candidate's skills in managing data types, cleaning data, and performing incremental updates, simulating a real-world scenario where data comes from third-party integrations and may contain errors.

Task Description:

1. **Data Extraction and Type Standardization to Staging Table**:
   - **Extract:** Use an SSIS package to extract data from a CSV file. Assume the source data in the CSV is in string format and may contain various errors.
   - **Transform:** Load the data into a staging table, converting string values to appropriate SQL data types (int, string, float, date, etc.). This step ensures all data is correctly formatted for further processing.
2. **Data Cleaning in Staging Table**:
   - **Clean:** After data is loaded into the staging table with correct data types, perform data cleaning operations including:
     - **Removing duplicates:** Identify and remove duplicate records based on specific criteria or key columns.
     - **Handling error records:** Isolate records that do not meet data quality standards (e.g., null values in non-nullable fields, incorrect formats) for review or correction.
3. **Incremental Load to Production Table**:
   - **Load:** Ensure the production table includes an ID column for unique record identification. Create a data flow from the staging table to the production table, differentiating new from existing records based on ID. Insert new records and update existing records.

Instructions:

- **Setup:** Begin by creating a public GitHub repository to store your SSIS solution. This repository will host all related files and the database backup.
- **Adapt and Overcome:** Address any discrepancies in the data or task details proactively. If the task description, DB schema, or CSV file contains errors or ambiguities, adjust and resolve these issues independently, applying your best judgment.
- **Code Quality:** Approach all coding and query writing as if developing for a production environment. Ensure that your code is clean, well-commented, and adheres to best practices. Optimize ETL processes, SQL queries, and scripts for performance, security, and maintainability.
- **Submit Your Solution:** Ensure all code and a backup of the database are pushed to your GitHub repository. Include a detailed README documenting your process, error handling, and outcomes. Respond to the email with a link to your GitHub repository, providing access to both your SSIS solution and the zipped database backup after the final SSIS execution.

For the Review Call:

- **Documentation:** Be prepared to discuss the configuration of your SSIS package, focusing on data type conversions, data cleaning logic, and the incremental load strategy.
- **Data Cleaning Rationale:** Explain your methods for data cleaning, including criteria for identifying and handling duplicates and error records.
- **Challenges:** Reflect on any challenges encountered and how they were resolved.
- **Package Adaptability:** Discuss how your package can accommodate changes in source data formats, database schema modifications, or scale to larger datasets, such as those containing 1 million records.

```sql
-- TODO: Replace {FirstName} and {LastName} with your actual first and last names before running this script
IF NOT EXISTS (SELECT * FROM sys.databases WHERE name = N'KoreAssignment_{firstName_lastName}')
```

```sql
BEGIN
    CREATE DATABASE [KoreAssignment_{FirstName_LastName}];
END
GO

USE [KoreAssignment_{FirstName_LastName}]
GO

-- Check and create stg schema if it does not exist
IF NOT EXISTS (SELECT * FROM sys.schemas WHERE name = N'stg')
BEGIN
    EXEC('CREATE SCHEMA stg');
END
GO

-- Check and create prod schema if it does not exist
IF NOT EXISTS (SELECT * FROM sys.schemas WHERE name = N'prod')
BEGIN
    EXEC('CREATE SCHEMA prod');
END
GO

-- Check and create stg.Users table if it does not exist
IF NOT EXISTS (SELECT * FROM sys.objects WHERE object_id = OBJECT_ID(N'stg.Users') AND type in (N'U'))
BEGIN
    CREATE TABLE stg.Users (
        StgID INT IDENTITY(1,1) PRIMARY KEY,
        UserID INT,
        FullName NVARCHAR(255),
        Age INT,
        Email NVARCHAR(255),
        RegistrationDate DATE,
        LastLoginDate DATE,
        PurchaseTotal FLOAT
    );
END
GO

-- Check and create prod.Users table if it does not exist
IF NOT EXISTS (SELECT * FROM sys.objects WHERE object_id = OBJECT_ID(N'prod.Users') AND type in (N'U'))
BEGIN
    CREATE TABLE prod.Users (
        ID INT IDENTITY(1,1) PRIMARY KEY,
        UserID INT,
        FullName NVARCHAR(255),
        Age INT,
        Email NVARCHAR(255),
        RegistrationDate DATE,
        LastLoginDate DATE,
        PurchaseTotal FLOAT,
        RecordLastUpdated DATETIME DEFAULT GETDATE()
    );
END
GO

```

1. **Data Insertion Only If Table Is Empty:** Use the following SQL script to insert initial data into the `prod.Users` table only if it is empty:

```sql
IF (SELECT COUNT(*) FROM prod.Users) = 0
```

```sql
BEGIN
    INSERT INTO prod.Users (UserID, FullName, Age, Email, RegistrationDate, LastLoginDate, PurchaseTotal)
    VALUES
    (101, 'John Doe', 30, 'johndoe@example.com', '2021-01-10', '2023-03-01', 150.00),
    (102, 'Jane Smith', 25, 'janesmith@example.com', '2020-05-15', '2023-02-25', 200.00),
    (103, 'Emily Johnson', 22, 'emilyjohnson@example.com', '2019-03-23', '2023-01-30', 120.50),
    (104, 'Michael Brown', 35, 'michaelbrown@example.com', '2018-07-18', '2023-02-20', 300.75),
    (105, 'Jessica Garcia', 28, 'jessicagarcia@example.com', '2022-08-05', '2023-02-18', 180.25),
    (106, 'David Miller', 40, 'davidmiller@example.com', '2017-12-12', '2023-02-15', 220.40),
    (107, 'Sarah Martinez', 33, 'sarahmartinez@example.com', '2018-11-30', '2023-02-10', 140.60),
    (108, 'James Taylor', 29, 'jamestaylor@example.com', '2019-06-22', '2023-02-05', 210.00),
    (109, 'Linda Anderson', 27, 'lindaanderson@example.com', '2021-04-16', '2023-01-25', 165.95),
    (110, 'Robert Wilson', 31, 'robertwilson@example.com', '2020-02-20', '2023-01-20', 175.00);
END

```

CSV Data:

```
UserID,FullName,Age,Email,RegistrationDate,LastLoginDate,PurchaseTotal
"101","John Doe","30","johndoe@example.com","2021-01-10","2023-03-01","150.00"
"111","Alice Johnson","","alicejohnson@example.com","2022-07-15","",""
"112","Bob Marley","27","bobmarley@example.com","2021-05-20","2023-02-28",""
"113","Cathy Smith","null","cathysmith@example.com","2019-10-12","2023-02-27","210.50"
"","Null User","25","","2020-12-01","2023-02-25","100.00"
"114","Derek Nowak","29","dereknowak@example.com","2020-03-17","2023-02-24","180.75"
"115","Eva Green","null","evagreen@example.com","2018-08-22","2023-02-23",""
"116","Frank Poe","34","frankpoe@example.com","2019-11-13","2023-02-22","190.40"
"101","John Doe","30","johndoe@example.com","2023-01-10","2023-03-01","200.00"
"117","George Kay","28","georgekay@example.com","2021-06-07","2023-02-21","170.00"
"118","Hanna Lux","32","hannalux@example.com","2019-02-18","2023-02-20","220.15"
"119","Ian Volt","26","ianvolt@example.com","2020-07-23","2023-02-19","130.00"
"120","Julia Nex","24","julianex@example.com","2022-01-12","2023-02-18","145.60"
"121","Kevin Yolt","abc","kevinyolt@example.com","2021-03-14","2023-02-17",""
"122","Lana Molt","31","lanamolt@example.com","2020-09-09","2023-02-16","155.30"
"123","Mike Dolt","33","mikedolt@example.com","2018-05-05","2023-02-15","205.00"
"124","Nina Colt","27","ninacolt@example.com","2019-07-29","2023-02-14","160.75"
"125","Oscar Holt","29","oscarholt@example.com","2022-11-11","2023-02-13","175.45"
"126","Patty Jolt","31","pattyjolt@example.com","2018-12-13","2023-02-12","185.90"
"127","Quincy Molt","34","quincymolt@example.com","2020-04-17","2023-02-11","195.25"
"128","Rita Bolt","36","ritabolt@example.com","2021-08-21","2023-02-10","210.50"
"112","Bob Marley","27","bobmarley@example.com","2021-05-20","2023-02-28",""
"129","Steve Jolt","38","stevejolt@example.com","2019-01-01","2023-02-09","225.75"
"130","Invalid Date","29","invaliddatetest@example.com","2022-02-30","2023-02-08","180.00"
"131","Special Characters Name","34","special$$name@example.com","2021-04-17","2023-02-07","200.50"
"132","Old Format","30","oldformat@example.com","07/05/2019","02/06/2023","150.50"
"133","Future Date","25","futuredate@example.com","2024-01-01","2025-01-02","160.00"
"134","Negative Age","-1","negativeage@example.com","2021-08-15","2023-02-04","130.00"
"135","Very Old Date","90","veryolddate@example.com","1920-01-01","2023-02-03","100.00"
"136","Extra Large Total","27","extralargetotal@example.com","2020-10-10","2023-02-02","1000000.00"
"137","Incorrect Email","28","notanemail","2021-11-11","2023-02-01","190.00"
"138","Duplicate ID","27","duplicateid@example.com","2019-03-15","2023-01-31","200.00"
"101","John Doe","30","johndoe@example.com","2024-01-10","2023-03-01","250.00"
```