```
pip install gspread google-auth pandas

Collecting gspread
  Downloading gspread-6.2.1-py3-none-any.whl.metadata (11 kB)
Collecting google-auth
  Downloading google_auth-2.41.1-py2.py3-none-any.whl.metadata (6.6
kB)
Requirement already satisfied: pandas in c:\users\pradeep kalluru\
anaconda3\lib\site-packages (2.2.2)
Collecting google-auth-oauthlib>=0.4.1 (from gspread)
  Downloading google_auth_oauthlib-1.2.2-py3-none-any.whl.metadata
(2.7 kB)
Requirement already satisfied: cachetools<7.0,>=2.0.0 in c:\users\
pradeep kalluru\anaconda3\lib\site-packages (from google-auth) (5.3.3)
Requirement already satisfied: pyasn1-modules>=0.2.1 in c:\users\
pradeep kalluru\anaconda3\lib\site-packages (from google-auth) (0.2.8)
Collecting rsa<5,>=3.1.4 (from google-auth)
  Downloading rsa-4.9.1-py3-none-any.whl.metadata (5.6 kB)
Requirement already satisfied: numpy>=1.26.0 in c:\users\pradeep
kalluru\anaconda3\lib\site-packages (from pandas) (1.26.4)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\
pradeep kalluru\anaconda3\lib\site-packages (from pandas)
(2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in c:\users\pradeep
kalluru\anaconda3\lib\site-packages (from pandas) (2024.1)
Requirement already satisfied: tzdata>=2022.7 in c:\users\pradeep
kalluru\anaconda3\lib\site-packages (from pandas) (2023.3)
Collecting requests-oauthlib>=0.7.0 (from google-auth-oauthlib>=0.4.1-
>gspread)
  Downloading requests_oauthlib-2.0.0-py2.py3-none-any.whl.metadata
(11 kB)
Requirement already satisfied: pyasn1<0.5.0,>=0.4.6 in c:\users\
pradeep kalluru\anaconda3\lib\site-packages (from pyasn1-
modules>=0.2.1->google-auth) (0.4.8)
Requirement already satisfied: six>=1.5 in c:\users\pradeep kalluru\
anaconda3\lib\site-packages (from python-dateutil>=2.8.2->pandas)
(1.16.0)
Collecting oauthlib>=3.0.0 (from requests-oauthlib>=0.7.0->google-
auth-oauthlib>=0.4.1->gspread)
  Downloading oauthlib-3.3.1-py3-none-any.whl.metadata (7.9 kB)
Requirement already satisfied: requests>=2.0.0 in c:\users\pradeep
kalluru\anaconda3\lib\site-packages (from requests-oauthlib>=0.7.0-
>google-auth-oauthlib>=0.4.1->gspread) (2.32.3)
Requirement already satisfied: charset-normalizer<4,>=2 in c:\users\
pradeep kalluru\anaconda3\lib\site-packages (from requests>=2.0.0-
>requests-oauthlib>=0.7.0->google-auth-oauthlib>=0.4.1->gspread)
(3.3.2)
Requirement already satisfied: idna<4,>=2.5 in c:\users\pradeep
kalluru\anaconda3\lib\site-packages (from requests>=2.0.0->requests-
oauthlib>=0.7.0->google-auth-oauthlib>=0.4.1->gspread) (3.7)
```

```
Requirement already satisfied: urllib3<3,>=1.21.1 in c:\users\pradeep
kalluru\anaconda3\lib\site-packages (from requests>=2.0.0->requests-
oauthlib>=0.7.0->google-auth-oauthlib>=0.4.1->gspread) (2.2.3)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\pradeep
kalluru\anaconda3\lib\site-packages (from requests>=2.0.0->requests-
oauthlib>=0.7.0->google-auth-oauthlib>=0.4.1->gspread) (2024.8.30)
Downloading gspread-6.2.1-py3-none-any.whl (59 kB)
Downloading google_auth-2.41.1-py2.py3-none-any.whl (221 kB)
Downloading google_auth_oauthlib-1.2.2-py3-none-any.whl (19 kB)
Downloading rsa-4.9.1-py3-none-any.whl (34 kB)
Downloading requests_oauthlib-2.0.0-py2.py3-none-any.whl (24 kB)
Downloading oauthlib-3.3.1-py3-none-any.whl (160 kB)
Installing collected packages: rsa, oauthlib, requests-oauthlib,
google-auth, google-auth-oauthlib, gspread
Successfully installed google-auth-2.41.1 google-auth-oauthlib-1.2.2
gspread-6.2.1 oauthlib-3.3.1 requests-oauthlib-2.0.0 rsa-4.9.1
Note: you may need to restart the kernel to use updated packages.

pip install pandas gspread sqlalchemy pyodbc

Requirement already satisfied: pandas in c:\users\pradeep kalluru\
anaconda3\lib\site-packages (2.2.2)
Requirement already satisfied: gspread in c:\users\pradeep kalluru\
anaconda3\lib\site-packages (6.2.1)
Requirement already satisfied: sqlalchemy in c:\users\pradeep kalluru\
anaconda3\lib\site-packages (2.0.34)
Requirement already satisfied: pyodbc in c:\users\pradeep kalluru\
anaconda3\lib\site-packages (5.1.0)
Requirement already satisfied: numpy>=1.26.0 in c:\users\pradeep
kalluru\anaconda3\lib\site-packages (from pandas) (1.26.4)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\
pradeep kalluru\anaconda3\lib\site-packages (from pandas)
(2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in c:\users\pradeep
kalluru\anaconda3\lib\site-packages (from pandas) (2024.1)
Requirement already satisfied: tzdata>=2022.7 in c:\users\pradeep
kalluru\anaconda3\lib\site-packages (from pandas) (2023.3)
Requirement already satisfied: google-auth>=1.12.0 in c:\users\pradeep
kalluru\anaconda3\lib\site-packages (from gspread) (2.41.1)
Requirement already satisfied: google-auth-oauthlib>=0.4.1 in c:\
users\pradeep kalluru\anaconda3\lib\site-packages (from gspread)
(1.2.2)
Requirement already satisfied: typing-extensions>=4.6.0 in c:\users\
pradeep kalluru\anaconda3\lib\site-packages (from sqlalchemy) (4.11.0)
Requirement already satisfied: greenlet!=0.4.17 in c:\users\pradeep
kalluru\anaconda3\lib\site-packages (from sqlalchemy) (3.0.1)
Requirement already satisfied: cachetools<7.0,>=2.0.0 in c:\users\
pradeep kalluru\anaconda3\lib\site-packages (from google-auth>=1.12.0-
>gspread) (5.3.3)
Requirement already satisfied: pyasn1-modules>=0.2.1 in c:\users\
```

```
pradeep kalluru\anaconda3\lib\site-packages (from google-auth>=1.12.0-
>gspread) (0.2.8)
Requirement already satisfied: rsa<5,>=3.1.4 in c:\users\pradeep
kalluru\anaconda3\lib\site-packages (from google-auth>=1.12.0-
>gspread) (4.9.1)
Requirement already satisfied: requests-oauthlib>=0.7.0 in c:\users\
pradeep kalluru\anaconda3\lib\site-packages (from google-auth-
oauthlib>=0.4.1->gspread) (2.0.0)
Requirement already satisfied: six>=1.5 in c:\users\pradeep kalluru\
anaconda3\lib\site-packages (from python-dateutil>=2.8.2->pandas)
(1.16.0)
Requirement already satisfied: pyasn1<0.5.0,>=0.4.6 in c:\users\
pradeep kalluru\anaconda3\lib\site-packages (from pyasn1-
modules>=0.2.1->google-auth>=1.12.0->gspread) (0.4.8)
Requirement already satisfied: oauthlib>=3.0.0 in c:\users\pradeep
kalluru\anaconda3\lib\site-packages (from requests-oauthlib>=0.7.0-
>google-auth-oauthlib>=0.4.1->gspread) (3.3.1)
Requirement already satisfied: requests>=2.0.0 in c:\users\pradeep
kalluru\anaconda3\lib\site-packages (from requests-oauthlib>=0.7.0-
>google-auth-oauthlib>=0.4.1->gspread) (2.32.3)
Requirement already satisfied: charset-normalizer<4,>=2 in c:\users\
pradeep kalluru\anaconda3\lib\site-packages (from requests>=2.0.0-
>requests-oauthlib>=0.7.0->google-auth-oauthlib>=0.4.1->gspread)
(3.3.2)
Requirement already satisfied: idna<4,>=2.5 in c:\users\pradeep
kalluru\anaconda3\lib\site-packages (from requests>=2.0.0->requests-
oauthlib>=0.7.0->google-auth-oauthlib>=0.4.1->gspread) (3.7)
Requirement already satisfied: urllib3<3,>=1.21.1 in c:\users\pradeep
kalluru\anaconda3\lib\site-packages (from requests>=2.0.0->requests-
oauthlib>=0.7.0->google-auth-oauthlib>=0.4.1->gspread) (2.2.3)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\pradeep
kalluru\anaconda3\lib\site-packages (from requests>=2.0.0->requests-
oauthlib>=0.7.0->google-auth-oauthlib>=0.4.1->gspread) (2024.8.30)
Note: you may need to restart the kernel to use updated packages.
```

```python
import gspread
import pandas as pd
import pyodbc
from google.oauth2.service_account import Credentials
from io import StringIO
import urllib.parse

# --- 1. GOOGLE SHEETS CONFIGURATION ---
GOOGLE_SHEET_NAME = "Sales_Live_Data"
CREDENTIALS_FILE = "salesautomation-474410-d0b68cf3a8a2.json"
# Define the scopes for Google Sheets and Drive access
SCOPES = [
    'https://www.googleapis.com/auth/spreadsheets.readonly',
    'https://www.googleapis.com/auth/drive.readonly'
]
```

```
# --- 2. SQL SERVER CONFIGURATION ---
# The connection string for your local SQL Server Express instance
SQL_SERVER_NAME = r"PRADEEP\SQLEXPRESS" # Use the raw string prefix
'r'
SQL_DATABASE_NAME = "SalesAnalytics"
SQL_TABLE_NAME = "Live_Data_Staging" # New table for the live data
ingestion

# You can use a specific driver. Common ones are 'ODBC Driver 17 for
SQL Server' or 'SQL Server'.
# Check your ODBC Data Sources (64-bit) for the exact name.
SQL_DRIVER = "ODBC Driver 17 for SQL Server"

# Use Windows Authentication (Trusted_Connection=yes) for local
server.
# If you use SQL Server Authentication, change to
UID=<user>;PWD=<password>
SQL_CONN_STRING =
f"DRIVER={{{SQL_DRIVER}}};SERVER={SQL_SERVER_NAME};DATABASE={SQL_DATAB
ASE_NAME};Trusted_Connection=yes;"

def run_automation():
    print("Starting data automation pipeline...")

    # --- 1. EXTRACT (Google Sheets) ---
    try:
        # Load credentials
        credentials =
Credentials.from_service_account_file(CREDENTIALS_FILE, scopes=SCOPES)
        gc = gspread.authorize(credentials)

        # Open the spreadsheet
        sh = gc.open(GOOGLE_SHEET_NAME)
        # Assuming your data is in the first worksheet
        worksheet = sh.worksheet(sh.sheet1.title)

        # Get all records as a list of lists (including header)
        data = worksheet.get_all_values()

        # Convert to a Pandas DataFrame
        if not data:
            print("Error: Google Sheet is empty.")
            return

        df = pd.DataFrame(data[1:], columns=data[0])
        print(f"Successfully extracted {len(df)} rows from Google
Sheets.")

    except Exception as e:
```

```
        print(f"Error during Google Sheets extraction: {e}")
        return

   # --- 2. TRANSFORM (Data Cleaning/Preparation) ---
    # Apply basic cleaning/type conversion here.

    # Drop any rows where all values are missing (e.g., empty rows
clients might add)
    df.dropna(how='all', inplace=True)

    if df.empty:
        print("DataFrame is empty after cleaning. Aborting load.")
        return

    # >>>>> START OF DATE CONVERSION BLOCK (FIXED INDENTATION) <<<<<
    try:
        # 1. Convert the column to datetime objects using the explicit
format
        #    NOTE: Ensure format='%d/%m/%Y' (DD/MM/YYYY) is correct
for your Google Sheet.
        df['OrderDate'] = pd.to_datetime(
            df['OrderDate'],
            format='%d/%m/%Y',
            errors='coerce'
        )

        # 2. Convert valid datetime objects to the SQL-friendly string
format (YYYY-MM-DD).
        df['OrderDate'] = df['OrderDate'].dt.strftime('%Y-%m-%d')

        # 3. Replace the resulting 'NaT' string (from invalid dates)
and empty strings with Python's None for SQL
        df = df.replace({'NaT': None, '': None})

        print("OrderDate successfully standardized to YYYY-MM-DD
format.")

    except KeyError:
        # This executes if the 'OrderDate' column header is not found
        print("Warning: 'OrderDate' column not found in Google Sheet
data. Skipping date conversion.")
    except Exception as e:
        # This handles any other unexpected error during date
transformation
        print(f"Error during date standardization: {e}")
        return
    # >>>>> END OF DATE CONVERSION BLOCK <<<<<
    # >>>>> END OF DATE CONVERSION BLOCK <<<<<

    # --- 3. LOAD (To SQL Server) ---
```

```
    CHUNK_SIZE = 10000  # Batch size for reliable insertion
    total_rows = len(df)

    try:
        # Establish connection to SQL Server
        conn = pyodbc.connect(SQL_CONN_STRING)
        cursor = conn.cursor()
        print("Connected to SQL Server.")

        # 1. Clear the staging table to load fresh data (TRUNCATE is
fast)
        truncate_sql = f"TRUNCATE TABLE {SQL_TABLE_NAME};"
        cursor.execute(truncate_sql)
        conn.commit()
        print(f"Cleared existing data from {SQL_TABLE_NAME}.")

        # 2. Build the dynamic INSERT statement
        columns = ', '.join([f'[{col}]' for col in df.columns])
        sql = f"INSERT INTO {SQL_TABLE_NAME} ({columns}) VALUES ({',
'.join(['?'] * len(df.columns))})"

        rows_loaded = 0

        # 3. Iterate over the DataFrame in chunks and insert each
batch
        for i in range(0, total_rows, CHUNK_SIZE):
            chunk_df = df.iloc[i:i + CHUNK_SIZE]

            # Prepare the list of tuples for insertion
            data_to_insert = [tuple(row) for row in chunk_df.values]

            # Execute batch insert
            cursor.executemany(sql, data_to_insert)
            conn.commit()

            rows_loaded += len(chunk_df)
            print(f"Loaded {rows_loaded} of {total_rows} rows...")

        print(f"Successfully loaded ALL {total_rows} rows into
{SQL_TABLE_NAME}.")

    except pyodbc.Error as ex:
        sqlstate = ex.args[0]
        print(f"Error during SQL Load: {sqlstate}. Details: {ex}")
    finally:
        if 'conn' in locals() and conn:
            conn.close()
            print("SQL Server connection closed.")
```

```
if __name__ == "__main__":
    run_automation()
```

```
Starting data automation pipeline...
Successfully extracted 250000 rows from Google Sheets.
OrderDate successfully standardized to YYYY-MM-DD format.
Connected to SQL Server.
Cleared existing data from Live_Data_Staging.
Loaded 10000 of 250000 rows...
Loaded 20000 of 250000 rows...
Loaded 30000 of 250000 rows...
Loaded 40000 of 250000 rows...
Loaded 50000 of 250000 rows...
Loaded 60000 of 250000 rows...
Loaded 70000 of 250000 rows...
Loaded 80000 of 250000 rows...
Loaded 90000 of 250000 rows...
Loaded 100000 of 250000 rows...
Loaded 110000 of 250000 rows...
Loaded 120000 of 250000 rows...
Loaded 130000 of 250000 rows...
Loaded 140000 of 250000 rows...
Loaded 150000 of 250000 rows...
Loaded 160000 of 250000 rows...
Loaded 170000 of 250000 rows...
Loaded 180000 of 250000 rows...
Loaded 190000 of 250000 rows...
Loaded 200000 of 250000 rows...
Loaded 210000 of 250000 rows...
Loaded 220000 of 250000 rows...
Loaded 230000 of 250000 rows...
Loaded 240000 of 250000 rows...
Loaded 250000 of 250000 rows...
Successfully loaded ALL 250000 rows into Live_Data_Staging.
SQL Server connection closed.
```