# 📘 Beginner-Friendly Guide: Transferring CSV Files from SFTP to Snowflake Using Python

This guide will help beginners and non-technical users create a Python-based data pipeline to transfer .csv files from an **SFTP server** to **Snowflake**, using a **Windows Server as the deployment environment**.

## ✅ Overview of the Process

1. Install Python and necessary packages
2. Set environment variables
3. Create a folder structure for the project
4. Write the Python script step by step
5. Test the script manually
6. Schedule the script to run automatically using Task Scheduler
7. Use Snowflake MERGE logic to prevent duplicates

## 🧰 Step 1: Install Python and Required Libraries

### ✅ Install Python

1. Go to <https://www.python.org/downloads>
2. Download and install Python 3.8 or later
3. During installation, check the box that says **“Add Python to PATH”**

### ✅ Install Required Python Libraries

Open Command Prompt and run:

pip install paramiko pandas snowflake-connector-python

If you have a requirements.txt file, run:

pip install -r requirements.txt

## 🗂️ Step 2: Create Project Folder Structure

Create a folder named C:\data\_pipeline\ with the following subfolders:

C:\data\_pipeline\  
├── pipeline\  
│ └── sftp\_to\_snowflake.py  
├── logs\  
└── requirements.txt

## 🔐 Step 3: Set Environment Variables in Windows

1. Press **Windows + S** and search for Environment Variables
2. Click **Edit the system environment variables**
3. In the dialog box, click **Environment Variables**
4. Under “System variables” or “User variables”, add the following:

### ✅ SFTP Settings

SFTP\_HOST=sftp.example.com  
SFTP\_PORT=22  
SFTP\_USER=my\_user  
SFTP\_PASSWORD=my\_password

### ✅ Snowflake Settings

SNOWFLAKE\_USER=your\_snowflake\_user  
SNOWFLAKE\_PASSWORD=your\_snowflake\_password  
SNOWFLAKE\_ACCOUNT=xy12345.us-east-1  
SNOWFLAKE\_WAREHOUSE=COMPUTE\_WH  
SNOWFLAKE\_DATABASE=MY\_DB  
SNOWFLAKE\_SCHEMA=PUBLIC

Click OK > OK > Apply to save changes.

## 🐍 Step 4: Write the Python Script

### 📄 File: C:\data\_pipeline\pipeline\sftp\_to\_snowflake.py

import os  
import paramiko  
import pandas as pd  
import snowflake.connector  
from snowflake.connector.pandas\_tools import write\_pandas  
import logging  
  
# === Setup logging ===  
LOG\_PATH = os.path.join(os.environ.get("USERPROFILE", "C:\\"), "data\_pipeline", "logs")  
os.makedirs(LOG\_PATH, exist\_ok=True)  
  
logging.basicConfig(  
 filename=os.path.join(LOG\_PATH, "transfer.log"),  
 level=logging.INFO,  
 format="%(asctime)s | %(levelname)s | %(message)s"  
)  
  
# === Read environment variables ===  
SFTP\_HOST = os.environ["SFTP\_HOST"]  
SFTP\_PORT = int(os.getenv("SFTP\_PORT", 22))  
SFTP\_USER = os.environ["SFTP\_USER"]  
SFTP\_PASSWORD = os.environ["SFTP\_PASSWORD"]  
  
SF\_USER = os.environ["SNOWFLAKE\_USER"]  
SF\_PASSWORD = os.environ["SNOWFLAKE\_PASSWORD"]  
SF\_ACCOUNT = os.environ["SNOWFLAKE\_ACCOUNT"]  
SF\_WAREHOUSE = os.environ["SNOWFLAKE\_WAREHOUSE"]  
SF\_DATABASE = os.environ["SNOWFLAKE\_DATABASE"]  
SF\_SCHEMA = os.environ["SNOWFLAKE\_SCHEMA"]  
  
REMOTE\_DIR = "/upload/"  
LOCAL\_TMP = os.environ.get("TEMP", "C:\\Temp")  
  
# === Connect to Snowflake ===  
def connect\_snowflake():  
 return snowflake.connector.connect(  
 user=SF\_USER,  
 password=SF\_PASSWORD,  
 account=SF\_ACCOUNT,  
 warehouse=SF\_WAREHOUSE,  
 database=SF\_DATABASE,  
 schema=SF\_SCHEMA  
 )  
  
# === Download CSVs from SFTP ===  
def download\_csvs\_from\_sftp():  
 transport = paramiko.Transport((SFTP\_HOST, SFTP\_PORT))  
 transport.connect(username=SFTP\_USER, password=SFTP\_PASSWORD)  
 sftp = paramiko.SFTPClient.from\_transport(transport)  
  
 downloaded\_files = []  
 for file\_attr in sftp.listdir\_attr(REMOTE\_DIR):  
 file\_name = file\_attr.filename  
 if not file\_name.endswith(".csv"):  
 continue  
  
 local\_path = os.path.join(LOCAL\_TMP, file\_name)  
 sftp.get(REMOTE\_DIR + file\_name, local\_path)  
 downloaded\_files.append((file\_name, local\_path))  
 sftp.rename(REMOTE\_DIR + file\_name, REMOTE\_DIR + "processed/" + file\_name)  
 logging.info(f"Downloaded and moved: {file\_name}")  
  
 sftp.close()  
 return downloaded\_files  
  
# === Upload to Snowflake ===  
def load\_csv\_to\_snowflake(file\_name, file\_path, conn):  
 table\_name = "stg\_application" # Always load to staging  
 df = pd.read\_csv(file\_path)  
 write\_pandas(conn, df, table\_name=table\_name, overwrite=False)  
 conn.cursor().execute(f"INSERT INTO file\_log (file\_name) VALUES ('{file\_name}')")  
 logging.info(f"✅ Uploaded {file\_name} to staging")  
  
# === Merge into Final Table ===  
def merge\_into\_application\_table(conn):  
 merge\_sql = """  
 MERGE INTO application AS tgt  
 USING stg\_application AS src  
 ON tgt.application\_id = src.application\_id  
 WHEN MATCHED THEN UPDATE SET  
 tgt.name = src.name,  
 tgt.date\_created = src.date\_created  
 WHEN NOT MATCHED THEN INSERT (application\_id, name, date\_created)  
 VALUES (src.application\_id, src.name, src.date\_created);  
 """  
 conn.cursor().execute(merge\_sql)  
 conn.cursor().execute("TRUNCATE TABLE stg\_application")  
 logging.info("✅ Merge and cleanup completed.")  
  
# === Run everything ===  
def run\_pipeline():  
 try:  
 conn = connect\_snowflake()  
 files = download\_csvs\_from\_sftp()  
 for file\_name, file\_path in files:  
 load\_csv\_to\_snowflake(file\_name, file\_path, conn)  
 os.remove(file\_path)  
 logging.info(f"Removed local file: {file\_path}")  
 merge\_into\_application\_table(conn)  
 conn.close()  
 logging.info("✅ Pipeline completed successfully.")  
 except Exception as e:  
 logging.error(f"❌ Pipeline failed: {str(e)}")  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 run\_pipeline()

## 🧪 Step 5: Test the Script Manually

1. Open Command Prompt
2. Run the script:

python C:\data\_pipeline\pipeline\sftp\_to\_snowflake.py

1. Check C:\data\_pipeline\logs\transfer.log for log output

## ⏱️ Step 6: Schedule the Script Using Task Scheduler

1. Open **Task Scheduler** (search in Start Menu)
2. Click **Create Basic Task**
3. Name it “SFTP to Snowflake Pipeline”
4. Choose your schedule (e.g., daily)
5. Action: **Start a Program**
   * Program/script: python
   * Add arguments: C:\data\_pipeline\pipeline\sftp\_to\_snowflake.py
6. Finish and click **Run** to test

## ✅ Step 7: Enable Deduplication with MERGE and Tracking

* Always upload to stg\_application
* Run MERGE to update/insert into final application table
* Use file\_log to track processed filenames
* Move files to processed/ folder to avoid reprocessing

## ✅ Done!

You now have an automated pipeline that:

* Connects to SFTP
* Downloads versioned CSVs
* Uploads to Snowflake staging
* Merges into your production table
* Prevents duplicates
* Runs on schedule from your Windows deployment server