Here's a Python script that reads a CSV file using Pandas and saves its content to a MySQL database hosted on a remote Windows PC.

### Prerequisites:

1. \*\*Install MySQL Server\*\* on the Windows machine.

2. \*\*Enable remote access\*\* to MySQL:

- Edit `my.cnf` or `my.ini` to allow connections:

```

bind-address = 0.0.0.0

```

Note:

**Why 0.0.0.0?**

By default, MySQL binds to 127.0.0.1, meaning it only accepts local connections (from the same machine).

* bind-address = 127.0.0.1 → Only local connections are allowed.
* bind-address = 192.168.X.X → Only allows connections from a specific IP.
* bind-address = 0.0.0.0 → Allows connections from any network interface (LAN, WAN, etc.).

\*\*\*\*\*\*\*\*\*\*\*\*

- Create a MySQL user with remote access:

```sql

CREATE USER 'your\_user'@'%' IDENTIFIED BY 'your\_password';

GRANT ALL PRIVILEGES ON your\_database.\* TO 'your\_user'@'%';

FLUSH PRIVILEGES;

```

- Allow MySQL through the Windows firewall.

3. \*\*Install required Python packages\*\* on the Linux PC:

```bash

pip install pandas pymysql

```

---

### Python Script:

```python

import pandas as pd

import pymysql

# MySQL Connection Details

DB\_HOST = "192.168.X.X" # Replace with Windows PC IP

DB\_PORT = 3306 # Default MySQL Port

DB\_USER = "your\_user"

DB\_PASSWORD = "your\_password"

DB\_NAME = "your\_database"

TABLE\_NAME = "your\_table"

CSV\_FILE = "data.csv" # Path to your CSV file

# Connect to MySQL

try:

conn = pymysql.connect(host=DB\_HOST, port=DB\_PORT, user=DB\_USER, password=DB\_PASSWORD, database=DB\_NAME)

cursor = conn.cursor()

print("Connected to MySQL successfully!")

except Exception as e:

print(f"Connection failed: {e}")

exit()

# Read CSV File

try:

df = pd.read\_csv(CSV\_FILE)

print("CSV file loaded successfully.")

except Exception as e:

print(f"Error reading CSV file: {e}")

exit()

# Create table if not exists (Modify column types as needed)

columns = ", ".join([f"{col} VARCHAR(255)" for col in df.columns])

create\_table\_query = f"CREATE TABLE IF NOT EXISTS {TABLE\_NAME} ({columns});"

cursor.execute(create\_table\_query)

# Insert Data

for \_, row in df.iterrows():

values = "', '".join(str(x) for x in row)

insert\_query = f"INSERT INTO {TABLE\_NAME} VALUES ('{values}');"

cursor.execute(insert\_query)

# Commit & Close

conn.commit()

cursor.close()

conn.close()

print("Data inserted successfully!")

```

---

### Notes:

✅ Replace `DB\_HOST`, `DB\_USER`, `DB\_PASSWORD`, `DB\_NAME`, and `TABLE\_NAME` accordingly.

✅ Ensure MySQL allows remote connections from the Linux PC.

✅ Modify column types if your data has different types (e.g., `INT`, `FLOAT`).

✅ If you get \*\*duplicate entry\*\* errors, consider adding `ON DUPLICATE KEY UPDATE`.

FYI

Let's break down the MySQL commands:

### \*\*1️⃣ CREATE USER 'your\_user'@'%' IDENTIFIED BY 'your\_password';\*\*

This command \*\*creates a new MySQL user\*\* who can connect from any IP address.

- \*\*`your\_user`\*\* → This is the username for authentication.

- \*\*`@'%'`\*\* → The \*\*`%` wildcard\*\* means the user can connect from \*\*any IP address\*\* (any host).

- If you want to allow access from only a specific IP (e.g., Linux PC at `192.168.1.100`), use:

```sql

CREATE USER 'your\_user'@'192.168.1.100' IDENTIFIED BY 'your\_password';

```

- \*\*`IDENTIFIED BY 'your\_password'`\*\* → Sets the password for the user.

---

### \*\*2️⃣ GRANT ALL PRIVILEGES ON your\_database.\* TO 'your\_user'@'%';\*\*

This command \*\*grants full access\*\* to the specified database.

- \*\*`ALL PRIVILEGES`\*\* → Gives the user permission to do everything in the database (`SELECT`, `INSERT`, `UPDATE`, `DELETE`, `CREATE`, etc.).

- \*\*`your\_database.\*`\*\* → Applies privileges to \*\*all tables\*\* (`\*`) inside `your\_database`.

- If you want to grant access to \*\*all databases\*\*, use:

```sql

GRANT ALL PRIVILEGES ON \*.\* TO 'your\_user'@'%';

```

- If you want \*\*read-only access\*\*, use:

```sql

GRANT SELECT ON your\_database.\* TO 'your\_user'@'%';

```

- \*\*`'your\_user'@'%'`\*\* → Grants these privileges to the user created earlier.

---

### \*\*3️⃣ FLUSH PRIVILEGES;\*\*

This command \*\*reloads MySQL’s privilege table\*\* so that the changes take effect immediately.

Without `FLUSH PRIVILEGES;`, MySQL may not apply the new permissions until a restart.

---

### \*\*Final Notes:\*\*

✅ This setup allows the user to access MySQL remotely from any IP.

✅ For better security, replace `%` with a specific IP (`192.168.1.100`).

✅ If needed, you can \*\*revoke\*\* privileges later with:

```sql

REVOKE ALL PRIVILEGES ON your\_database.\* FROM 'your\_user'@'%';