Foundational Data Engineering Practice Pack Using PostgreSQL + Python.

🎯 What Will be Covered (Core DE Skills)

Skill	Covered
Data Extraction	✅
File Export (CSV/Excel)	✅
Large File Handling	✅
Data Ingestion	✅
Stored Procedure Execution	✅
Table Maintenance	✅
Parameterized Query	✅

✅ REQUIRED LIBRARIES

Install these first:

```
pip install psycopg2-binary pandas sqlalchemy openpyxl xlsxwriter
```

What they are used for:

- `psycopg2` → Direct PostgreSQL connection
- `sqlalchemy` → Engine management
- `pandas` → Data handling
- `openpyxl` → Excel writing
- `xlsxwriter` → Advanced Excel handling

DATABASE TABLES USED:

We'll use:

- **`customers`**
- **`accounts`**
- **`transactions`**

🧠 TASK 1

✅ TASK QUESTION

Write a Python script to export all records from the `customers` table in PostgreSQL into a CSV file named `customers_export.csv`.
Name the file **python_full_script_export_postgres_to_csv.py**

✅ FULL SCRIPT – EXPORT POSTGRES TABLE TO CSV

```
import psycopg2
import csv

DB_CONFIG = {
    "host": "localhost",
    "port": "5432",
    "database": "your_database",
    "user": "your_user",
    "password": "your_password"
}

OUTPUT_FILE = "customers_export.csv"

def export_table_to_csv():
    try:
        conn = psycopg2.connect(**DB_CONFIG)
        cursor = conn.cursor()

        query = "SELECT * FROM customers;"
        cursor.execute(query)

        colnames = [desc[0] for desc in cursor.description]

        with open(OUTPUT_FILE, "w", newline="", encoding="utf-8") as f:
            writer = csv.writer(f)
            writer.writerow(colnames)
            writer.writerows(cursor.fetchall())
        print("✅ Customers exported successfully")
    except Exception as e:
        print("❌ Error:", e)
    finally:
        cursor.close()
        conn.close()

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

🧠 TASK 2

✅ TASK QUESTION

Export all records from the `transactions` table into an Excel file.
If the number of rows exceeds 1,000,000, split the data into multiple sheets.
Name the file as: **python_full_script_export_to_excel_with_split.py**

✅ FULL SCRIPT – EXPORT TO EXCEL WITH SHEET SPLIT

```
import pandas as pd
from sqlalchemy import create_engine

MAX_ROWS_PER_SHEET = 1_000_000
OUTPUT_FILE = "transactions_export.xlsx"

engine = create_engine(

"postgresql+psycopg2://your_user:your_password@localhost:5432/your_database"
)


def export_transactions_to_excel():
    df = pd.read_sql("SELECT * FROM transactions", engine)

    with pd.ExcelWriter(OUTPUT_FILE, engine="xlsxwriter") as writer:
        total_rows = len(df)
        sheet_number = 1

        for start in range(0, total_rows, MAX_ROWS_PER_SHEET):
            end = start + MAX_ROWS_PER_SHEET
            df_chunk = df.iloc[start:end]

            sheet_name = f"Sheet{sheet_number}"
            df_chunk.to_excel(writer, sheet_name=sheet_name, index=False)

            sheet_number += 1
    print("Transactions exported to Excel successfully")
```

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

🧠 **TASK 3**

✅ **TASK QUESTION**

Create a new PostgreSQL table named `customers_stage` and ingest records from a CSV file into it using Python.
Name the file as: **python_full_script_load_csv_into_postgres_table.py**

✅ **FULL SCRIPT – LOAD CSV INTO POSTGRES TABLE**

```
import pandas as pd
from sqlalchemy import create_engine

CSV_FILE = "customers_export.csv"
TABLE_NAME = "customers_stage"

engine = create_engine(
  "postgresql+psycopg2://your_user:your_password@localhost:5432/your_database"
)
def load_csv_to_postgres():
    df = pd.read_csv(CSV_FILE)

    df.to_sql(
        name=TABLE_NAME,
        con=engine,
        if_exists="replace",  # change to append if needed
        index=False,
        chunksize=5000
    )
    print("CSV data loaded into PostgreSQL successfully")
if __name__ == "__main__":
    load_csv_to_postgres()
```

🧠 TASK 4

✅ TASK QUESTION

Write a Python script to call a stored procedure named `sp_update_account_balance()` in PostgreSQL. Name the file as: **python+full_script_call_stored_procedure.py**

✅ FULL SCRIPT – CALL STORED PROCEDURE

```
import psycopg2

DB_CONFIG = {
    "host": "localhost",
    "port": "5432",
    "database": "your_database",
    "user": "your_user",
    "password": "your_password"
}

def call_procedure():
    try:
        conn = psycopg2.connect(**DB_CONFIG)
        cursor = conn.cursor()

        cursor.callproc("sp_update_account_balance")

        conn.commit()

        print("✅ Stored procedure executed successfully")

    except Exception as e:
        print("❌ Error:", e)
        conn.rollback()

    finally:
        cursor.close()
        conn.close()


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

🧠 **TASK 5**

✅ **TASK QUESTION**

Write a Python script to:

1. Truncate the `transactions` table
2. Insert records for a specific date (e.g., '2025-01-01') from `transactions_backup` table

Name the file as: python_full_script_truncate_insert_by_date

✅ **FULL SCRIPT – TRUNCATE + INSERT BY DATE**

```
import psycopg2

TARGET_DATE = "2025-01-01"

DB_CONFIG = {
    "host": "localhost",
    "port": "5432",
    "database": "your_database",
    "user": "your_user",
    "password": "your_password"
}


def truncate_and_insert():
    try:
        conn = psycopg2.connect(**DB_CONFIG)
        cursor = conn.cursor()

        # Truncate table
        cursor.execute("TRUNCATE TABLE transactions;")

        # Insert specific date records
        insert_query = """
            INSERT INTO transactions
            SELECT *
            FROM transactions_backup
            WHERE tran_date = %s;
        """

        cursor.execute(insert_query, (TARGET_DATE,))
        conn.commit()

        print(f"✅ Data inserted for date {TARGET_DATE}")

    except Exception as e:
        print("❌ Error:", e)
        conn.rollback()

    finally:
        cursor.close()
        conn.close()

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