

# Data Ingestion Into PostgreSQL

1. Load CSV File Into Dataframe
2. Establish PostgreSQL Database Connection
3. Create PostgreSQL Table
4. Load Dataframe Into PostgreSQL Table

```
In [23]: import pandas as pd
import numpy as np
import psycopg2
```

## Load CSV File Into Dataframe

```
In [24]: df = pd.read_csv('/Users/michaeltyus/Desktop/sales_records.csv')
df.head()
```

```
Out[24]:
```

	Region	Country	Item Type	Sales Channel	Order Priority	Order Date	Order ID	Ship Date	Units Sold
0	Sub-Saharan Africa	Chad	Office Supplies	Online	L	1/27/2011	292494523	2/12/2011	4484
1	Europe	Latvia	Beverages	Online	C	12/28/2015	361825549	1/23/2016	1075
2	Middle East and North Africa	Pakistan	Vegetables	Offline	C	1/13/2011	141515767	2/1/2011	6515
3	Sub-Saharan Africa	Democratic Republic of the Congo	Household	Online	C	9/11/2012	500364005	10/6/2012	7683
4	Europe	Czech Republic	Beverages	Online	C	10/27/2015	127481591	12/5/2015	3491

```
In [25]: df.shape
```

```
Out[25]: (10000, 14)
```

```
In [26]: df.columns
```

```
Out[26]: Index(['Region', 'Country', 'Item Type', 'Sales Channel', 'Order Priority',
              'Order Date', 'Order ID', 'Ship Date', 'Units Sold', 'Unit Price',
              'Unit Cost', 'Total Revenue', 'Total Cost', 'Total Profit'],
              dtype='object')
```

## Establish PostgreSQL Database Connection

```
In [27]: connection = psycopg2.connect(user="postgres",
                                         password="*****",
                                         host="localhost",
                                         port="5432",
                                         database="sales_data")

cursor = connection.cursor()
print('Database Connection Successful...')
```

Database Connection Successful...

## Create PostgreSQL Table

```
In [28]: # Make dataframe column names lowercase and replace spaces
# in column names with underscores.
df.columns = df.columns.str.lower()
df.columns = df.columns.str.replace(" ", "_")
df.columns
```

```
Out[28]: Index(['region', 'country', 'item_type', 'sales_channel', 'order_priority',
               'order_date', 'order_id', 'ship_date', 'units_sold', 'unit_price',
               'unit_cost', 'total_revenue', 'total_cost', 'total_profit'],
              dtype='object')
```

```
In [29]: # Get column data types.
df.dtypes
```

```
Out[29]: region          object
country          object
item_type         object
sales_channel     object
order_priority    object
order_date        object
order_id          int64
ship_date         object
units_sold        int64
unit_price        float64
unit_cost         float64
total_revenue     float64
total_cost        float64
total_profit      float64
dtype: object
```

```
In [30]: # Create Pandas/PostgreSQL data type mapping dictionary
# and create the PostgreSQL table DDL.
dt_map = {'object': 'varchar',
          'int64': 'int',
          'float64': 'float'}
col_ddl = ", ".join("{} {}".format(x, y) for (x, y) \
                  in zip(df.columns, df.dtypes.replace(dt_map)))
ddl_sql = "CREATE TABLE sales.sales_info (" + col_ddl + ")"
print(ddl_sql)
```

```
CREATE TABLE sales.sales_info (region varchar, country varchar, item_type varc
har, sales_channel varchar, order_priority varchar, order_date varchar, order_
id int, ship_date varchar, units_sold int, unit_price float, unit_cost float,
total_revenue float, total_cost float, total_profit float)
```

```
In [31]: # Create the PostgreSQL table
cursor.execute(ddl_sql)
connection.commit()
print('Database Table Created...')
cursor.execute('SELECT COUNT(*) FROM sales.sales_info')
cursor.fetchall()
```

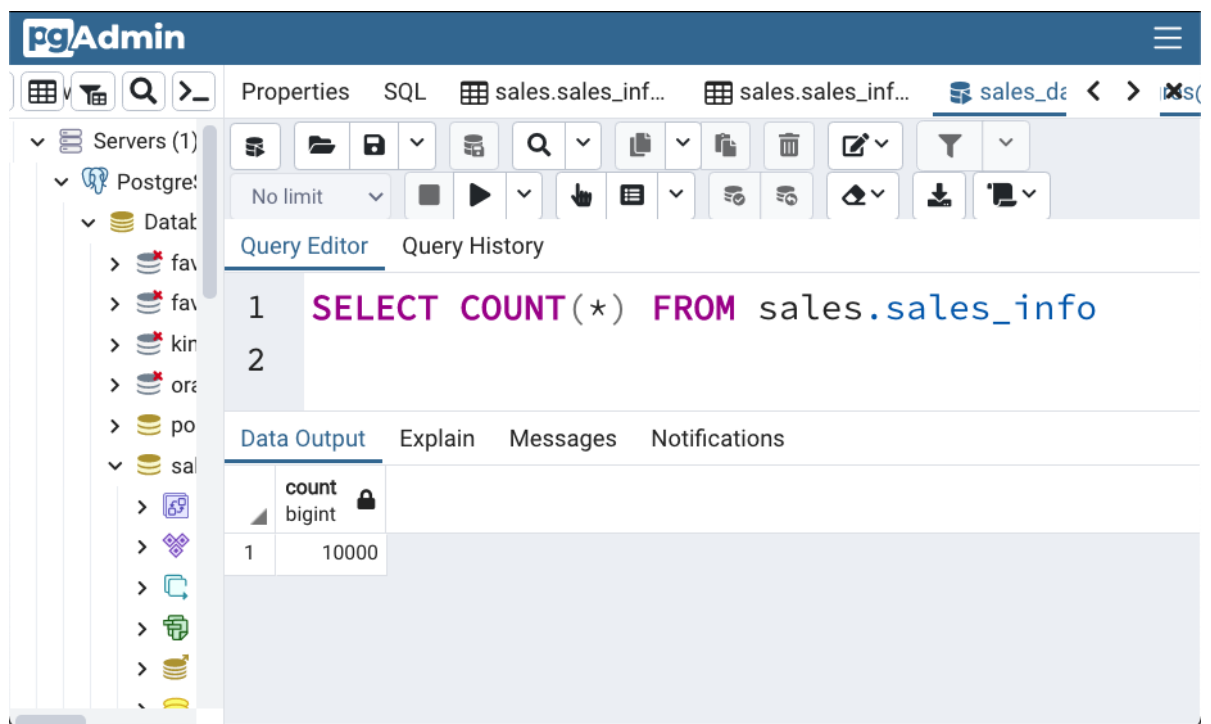
Database Table Created...

Out[31]: [(0,)]

## Load Dataframe Into PostgreSQL Table

```
In [32]: records = [tuple(x) for x in df.to_numpy()]
sql_dml = """INSERT INTO sales.sales_info (region, country, \
item_type, sales_channel, order_priority, order_date, \
order_id, ship_date, units_sold, unit_price, unit_cost, \
total_revenue, total_cost, total_profit) \
values(%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s)"""
cursor.executemany(sql_dml, records)
connection.commit()
cursor.execute('SELECT COUNT(*) FROM sales.sales_info')
cursor.fetchall()
```

Out[32]: [(10000,)]



```
In [33]: connection.close()
print('Database Connection Closed...')
```

Database Connection Closed...