

## COFFEE SALES ETL PORTFOLIO PROJECT

This notebook demonstrates a mini ETL pipeline:

1. Load data from .csv file from Kaggle
2. Clean data using Python/Pandas
3. Upload cleaned data to a PostgreSQL database
4. Run SQL queries
5. Create visualizations using Python/Matplotlib

## In the next step, we load the Kaggle\_CoffeeSales.csv file downloaded from Kaggle into a Pandas dataframe.

We check data types, handle missing values if any and remove duplicates from rows

```
In [ ]: import pandas as pd
df = pd.read_csv("Kaggle_CoffeeSales.csv")
df.head()
```

```
In [3]: df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 262 entries, 0 to 261
Data columns (total 5 columns):
 #   Column          Non-Null Count  Dtype
---  -
 0   date            262 non-null   object
 1   datetime        262 non-null   object
 2   cash_type       262 non-null   object
 3   money           262 non-null   float64
 4   coffee_name     262 non-null   object
dtypes: float64(1), object(4)
memory usage: 10.4+ KB
```

```
In [5]: df['date'] = pd.to_datetime(df['date'])
df['datetime'] = pd.to_datetime(df['datetime'])
```

```
In [7]: df.dtypes
```

```
Out[7]: date            datetime64[ns]
datetime        datetime64[ns]
cash_type              object
money                 float64
coffee_name          object
dtype: object
```

```
In [9]: df.duplicated().sum()
```

```
Out[9]: 2
```

```
In [11]: df = df.drop_duplicates()
```

```
In [14]: df.duplicated().sum()
```

Out[14]: 0

```
In [16]: df[df['money'] <= 0]
```

```
Out[16]:    date    datetime    cash_type    money    coffee_name
```

```
In [18]: df.isnull().sum()
```

```
Out[18]:  date            0
          datetime       0
          cash_type      0
          money          0
          coffee_name     0
          dtype: int64
```

```
In [20]: df.describe()
```

Out[20]:		date	datetime	money
count		260	260	260.000000
mean	2025-02-28 13:12:00	2025-03-01 04:29:05.738461440		26.307692
min	2025-02-08 00:00:00	2025-02-08 14:26:04		15.000000
25%	2025-02-17 00:00:00	2025-02-17 21:14:14.750000128		25.000000
50%	2025-02-28 00:00:00	2025-02-28 13:21:28.500000		27.000000
75%	2025-03-12 00:00:00	2025-03-12 10:45:04.500000		29.000000
max	2025-03-23 00:00:00	2025-03-23 21:23:11		33.000000
std		NaN	NaN	4.265053

## Uploading data to PostgreSQL

The cleaned dataset is uploaded to the PostgreSQL "Coffee\_Sales" table to enable running SQL query commands

```
In [1]: import psycopg2
import pandas as pd

conn = psycopg2.connect(
    dbname="postgres",
    user="postgres",
    password="XXXXXXXXXX",
    host="localhost",
    port="5432"
)

print("Connection successful")
```

Connection successful

```
In [5]: from sqlalchemy import create_engine
import pandas as pd

df = pd.read_csv("Kaggle_CoffeeSales.csv")

engine = create_engine("postgresql+psycopg2://postgres:szarvasvilmoS1@localhost:

df.to_sql("Coffee_Sales", engine, if_exists="replace", index=False)

print("Table successfully created in PostgreSQL")
```

Table successfully created in PostgreSQL

## SQL queries and visualization

- Daily revenue line chart
- Revenue by coffee type bar chart
- Revenue and transaction count by payment type bar and pie charts

```
In [5]: import pandas as pd
import matplotlib.pyplot as plt
from sqlalchemy import create_engine
```

```
In [17]: engine = create_engine("postgresql+psycopg2://postgres:szarvasvilmoS1@localhost:
```

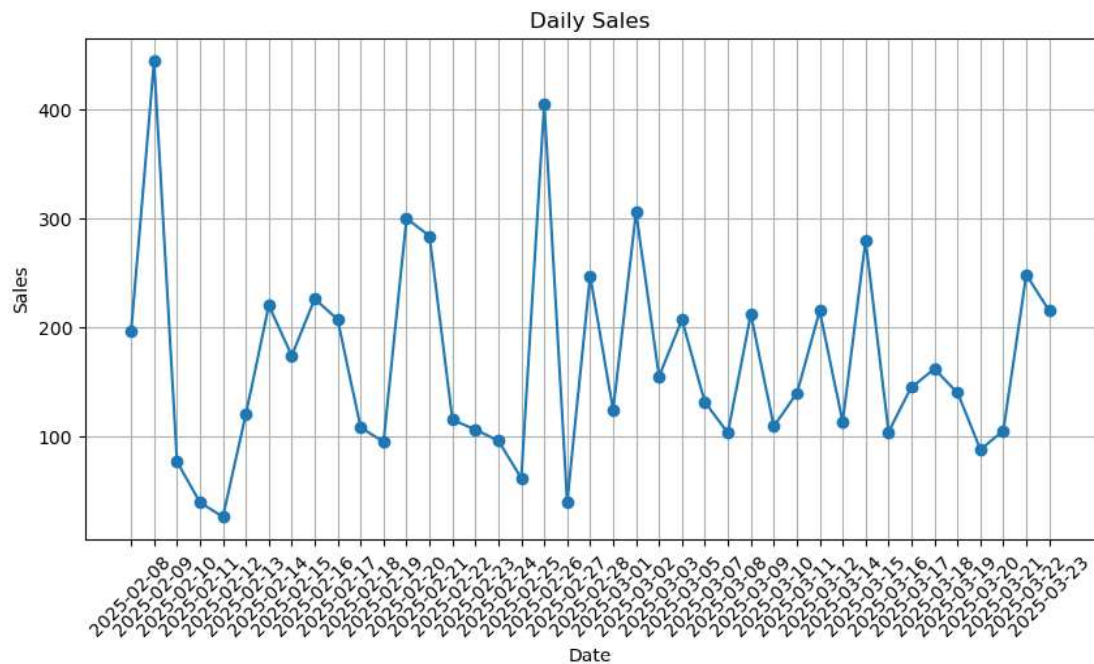
```
In [19]: daily_sales_sql = """
select date, sum(money) as total_daily_sales from "Coffee_Sales"
group by date
order by date;
"""

daily_sales = pd.read_sql(daily_sales_sql, engine)
daily_sales.head()
```

```
Out[19]:
```

	date	total_daily_sales
0	2025-02-08	196.0
1	2025-02-09	444.0
2	2025-02-10	77.0
3	2025-02-11	40.0
4	2025-02-12	27.0

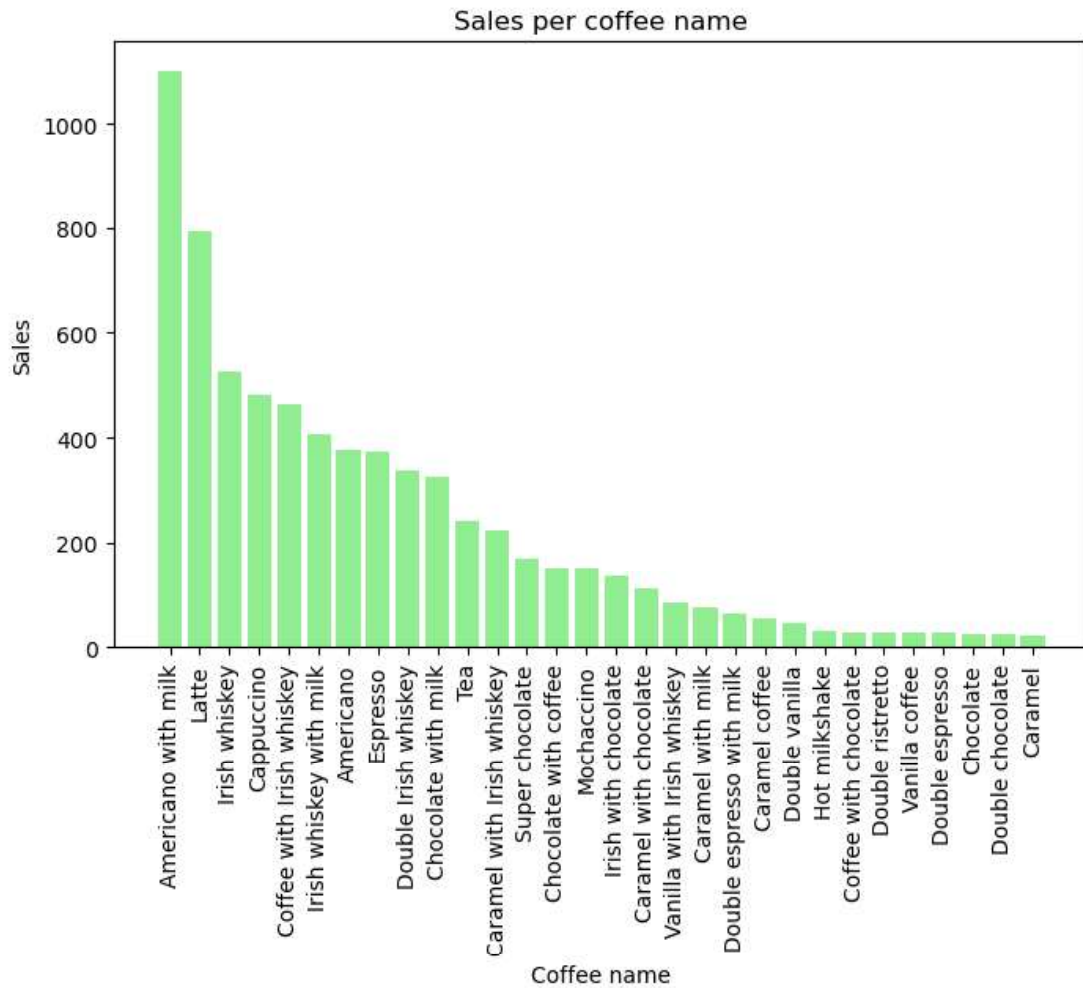
```
In [23]: plt.figure(figsize=(10,5))
plt.plot(daily_sales['date'], daily_sales['total_daily_sales'], marker = 'o')
plt.title('Daily Sales')
plt.xlabel('Date')
plt.ylabel('Sales')
plt.xticks(rotation=45)
plt.grid(True)
plt.show()
```



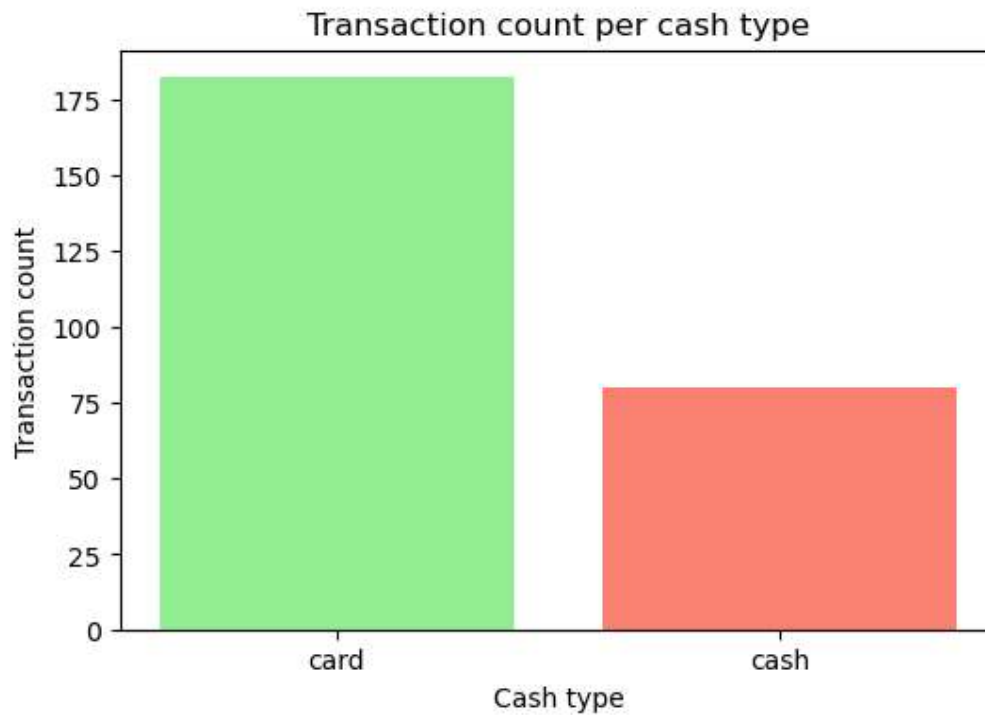
```
In [35]: coffee_sales_sql = """
select coffee_name, SUM(money) as total_sales from "Coffee_Sales"
group by coffee_name
order by total_sales desc;
"""

coffee_sales = pd.read_sql(coffee_sales_sql, engine)

plt.figure(figsize=(8,5))
plt.bar(coffee_sales['coffee_name'], coffee_sales['total_sales'], color='lightgray')
plt.title('Sales per coffee name')
plt.xlabel('Coffee name')
plt.ylabel('Sales')
plt.xticks(rotation=90)
plt.show()
```



```
In [37]: plt.figure(figsize=(6,4))
plt.bar(cash_sales['cash_type'], cash_sales['transaction_count'], color='lightg
plt.title('Transaction count per cash type')
plt.xlabel('Cash type')
plt.ylabel('Transaction count')
plt.show()
```



```
In [39]: cash_sales_sql = """
select cash_type, COUNT (*) as transaction_count, SUM(money) as total_revenue f
group by cash_type;
"""

cash_sales = pd.read_sql(cash_sales_sql, engine)

plt.figure(figsize=(6,6))
plt.pie(cash_sales['total_revenue'], labels=cash_sales['cash_type'], autopct='%1
plt.title(')
```

## Sales per cash type

