[1]:
```python
import pandas as pd
import mysql.connector
import os

# List of CSV files and their corresponding table names
csv_files = [
    ('customers.csv', 'customers'),
    ('orders.csv', 'orders'),
    ('sellers.csv', 'sellers'),
    ('products.csv', 'products'),
    ('geolocation.csv', 'geolocation'),
    ('payments.csv', 'payments'),
    ('order_items.csv','order_items')
    # Added payments.csv for specific handling
]

# Connect to the MySQL database
conn = mysql.connector.connect(
    host='localhost',
    user='root',
    password='@Rohit4545',
    database='ecommerce'
)
cursor = conn.cursor()

# Folder containing the CSV files
folder_path = 'C:/Users/rohis/OneDrive/Desktop/ecommers'

def get_sql_type(dtype):
    if pd.api.types.is_integer_dtype(dtype):
        return 'INT'
    elif pd.api.types.is_float_dtype(dtype):
        return 'FLOAT'
    elif pd.api.types.is_bool_dtype(dtype):
        return 'BOOLEAN'
    elif pd.api.types.is_datetime64_any_dtype(dtype):
        return 'DATETIME'
    else:
        return 'TEXT'

for csv_file, table_name in csv_files:
    file_path = os.path.join(folder_path, csv_file)

    # Read the CSV file into a pandas DataFrame
    df = pd.read_csv(file_path)

    # Replace NaN with None to handle SQL NULL
    df = df.where(pd.notnull(df), None)

    # Debugging: Check for NaN values
    print(f"Processing {csv_file}")
    print(f"NaN values before replacement:\n{df.isnull().sum()}\n")

    # Clean column names
    df.columns = [col.replace(' ', '_').replace('-', '_').replace('.', '_') for col in df.columns]

    # Generate the CREATE TABLE statement with appropriate data types
    columns = ', '.join([f'`{col}` {get_sql_type(df[col].dtype)}' for col in df.columns])
    create_table_query = f'CREATE TABLE IF NOT EXISTS `{table_name}` ({columns})'
    cursor.execute(create_table_query)

    # Insert DataFrame data into the MySQL table
    for _, row in df.iterrows():
        # Convert row to tuple and handle NaN/None explicitly
        values = tuple(None if pd.isna(x) else x for x in row)
        sql = f"INSERT INTO `{table_name}` ({', '.join(['`' + col + '`' for col in df.columns])}) VALUES ({', '.join(['%s'] * len(row))})"
        cursor.execute(sql, values)

    # Commit the transaction for the current CSV file
    conn.commit()

# Close the connection
conn.close()
```

```python
# Close the connection
conn.close()
```

```
Processing customers.csv
NaN values before replacement:
customer_id                 0
customer_unique_id          0
customer_zip_code_prefix    0
customer_city               0
customer_state              0
dtype: int64

Processing orders.csv
NaN values before replacement:
order_id                            0
customer_id                         0
order_status                        0
order_purchase_timestamp            0
order_approved_at                 160
order_delivered_carrier_date     1783
order_delivered_customer_date    2965
order_estimated_delivery_date       0
dtype: int64

Processing sellers.csv
NaN values before replacement:
seller_id                 0
seller_zip_code_prefix    0
seller_city               0
seller_state              0
dtype: int64

Processing products.csv
NaN values before replacement:
product_id                      0
product category              610
product_name_length           610
product_description_length    610
product_photos_qty            610
```

```python
[2]: import pandas as pd
     import matplotlib.pyplot as plt
     import seaborn as sns
     import mysql.connector
     import numpy as np

     db = mysql.connector.connect(host = "localhost",
                                  username = "root",
                                  password = "@Rohit4545",
                                  database = "ecommerce")
     cur = db.cursor()
```

List all unique cities where customers are located.

```python
[3]: query = """ select distinct(customer_city) from customers """

     cur.execute(query)

     data = cur.fetchall()

     data
```

```
[3]: [('franca',),
      ('sao bernardo do campo',),
      ('sao paulo',),
      ('mogi das cruzes',),
      ('campinas',),
      ('jaragua do sul',),
      ('timoteo',),
      ('curitiba',),
      ('belo horizonte',),
      ('montes claros',),
      ('rio de janeiro',),
      ('lencois paulista',),
```

Find the total sales per category.

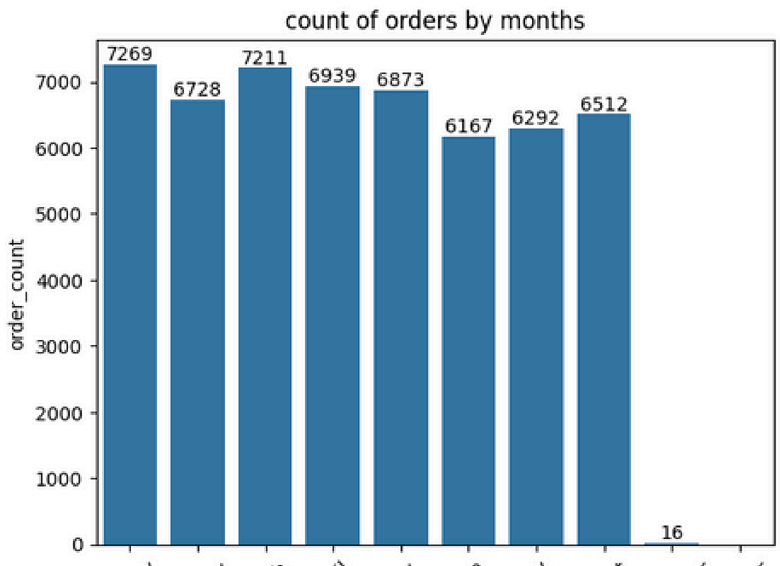
```
[8]: query = """ select products.product_category category,
     round(sum(payments.payment_value),2) sales
     from products join order_items
     on products.product_id = order_items.product_id
     join payments
     on payments.order_id = order_items.order_id
     group by category
     """

     cur.execute(query)

     data = cur.fetchall()

     df = pd.DataFrame(data, columns = ["Category", "Sales"])

     df.head()
```

[8]:

	Category	Sales
0	perfumery	4053909.28
1	Furniture Decoration	11441411.13
2	telephony	3895056.41
3	bed table bath	13700429.37
4	automotive	6818354.65

Calculate the percentage of orders that were paid in installments

```
[16]: query = """ select monthname(order_purchase_timestamp) months, count(order_id) order_count
      from orders where year(order_purchase_timestamp) = 2018
      group by months"""

      cur.execute(query)

      data = cur.fetchall()

      df = pd.DataFrame(data, columns = ["months","order_count"])

      o = ["January","February","March","April","May","June","July","August","September","Octuber"]

      ax = sns.barplot(x = df["months"],y = df["order_count"], data = df, order =o)
      plt.xticks(rotation = 45)
      ax.bar_label(ax.containers[0])
      plt.title("count of orders by months")

      plt.show()
```



Calculate the moving average of order values for each customer over their order history.

[42]:

```python
query = """ select customer_id,order_purchase_timestamp, payment,
avg(payment) over(partition by customer_id order by order_purchase_timestamp
rows between 2 preceding and current row) as mov_avg
from
(select orders.customer_id, orders.order_purchase_timestamp,
payments.payment_value as payment
from payments join orders
on payments.order_id = orders.order_id) as a"""

cur.execute(query)

data = cur.fetchall()

df = pd.DataFrame(data, columns = ["customer_id", "order_purchase_timestamp", "payment", "moving_avg"])
df.head(10)
```

[42]:

	customer_id	order_purchase_timestamp	payment	moving_avg
0	00012a2ce6f8dcda20d059ce98491703	2017-11-14 16:08:26	114.74	114.739998
1	000161a058600d5901f007fab4c27140	2017-07-16 09:40:32	67.41	67.410004
2	0001fd6190edaaf884bcaf3d49edf079	2017-02-28 11:06:43	195.42	195.419998
3	0002414f95344307404f0ace7a26f1d5	2017-08-16 13:09:20	179.35	179.350006
4	000379cdec625522490c315e70c7a9fb	2018-04-02 13:42:17	107.01	107.010002
5	0004164d20a9e969af783496f3408652	2017-04-12 08:35:12	71.80	71.800003
6	000419c5494106c306a97b5635748086	2018-03-02 17:47:40	49.40	49.400002
7	00046a560d407e99b969756e0b10f282	2017-12-18 11:08:30	166.59	166.589996
8	00050bf6e01e69d5c0fd612f1bcfb69c	2017-09-17 16:04:44	85.23	85.230003
9	000598caf2ef4117407665ac33275130	2018-08-11 12:14:35	1255.71	1255.709961