```
Task1 Importing the "telecom_dataset.csv" data and
combining all data files into one consolidated
dataframe.
```

Task1 Importing the "telecom_dataset.csv" data and combining all data files into one consolidated dataframe.

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
# import pandas as pd
import pandas as pd

# Takes the file's folder
filepath = "/content/drive/MyDrive/telecom_dataset.csv";

# read the CSV file
df = pd.read_csv(filepath)

# print the first five rows
print(df.head())
```

```
  state  account length  area code phone number international plan  \
0    KS             128        415     382-4657                 no
1    OH             107        415     371-7191                 no
2    NJ             137        415     358-1921                 no
3    OH              84        408     375-9999                yes
4    OK              75        415     330-6626                yes

  voice mail plan  number vmail messages  total day minutes  total day calls  \
0             yes                     25              265.1              110
1             yes                     26              161.6              123
2              no                      0              243.4              114
3              no                      0              299.4               71
4              no                      0              166.7              113

   Predicted total day calls  ...  total eve calls  total eve charge  \
0                        100  ...               99             16.78
1                        125  ...              103             16.62
2                        113  ...              110             10.30
3                         70  ...               88              5.26
4                        111  ...              122             12.61

   total night minutes  total night calls  total night charge  \
0                244.7                 91               11.01
1                254.4                103               11.45
2                162.6                104                7.32
3                196.9                 89                8.86
4                186.9                121                8.41

   total intl minutes  total intl calls  total intl charge  \
0                10.0                 3               2.70
1                13.7                 3               3.70
2                12.2                 5               3.29
3                 6.6                 7               1.78
4                10.1                 3               2.73

   customer service calls  churn
0                       1  False
1                       1  False
2                       0  False
3                       2  False
4                       3  False

[5 rows x 22 columns]
```

Task 2 How do you analyze“telecom_dataset.csv” data inNumPy and Pandaslibraries? Explain how you would handle missing data during data analysis using Pandas. Can you calculate and provide examples of commonly used statistics in data analysis, such as mean, median, standard deviation, minimum, maximum, and quantiles, using Pandas and NumPy?

```
# Check for missing values
missing_values = df.isna().sum()
print("Missing Values:\n", missing_values)
# Check for duplicate rows
print("\nDuplicate Rows:")
print(df.duplicated().sum())
```

```
Missing Values:
 state                         0
account length                 0
area code                      0
phone number                   0
international plan             0
voice mail plan                0
number vmail messages          0
total day minutes              0
total day calls                0
Predicted total day calls      0
total day charge               0
total eve minutes              0
total eve calls                0
total eve charge               0
total night minutes            0
total night calls              0
total night charge             0
total intl minutes             0
total intl calls               0
total intl charge              0
customer service calls         0
churn                          0
dtype: int64

Duplicate Rows:
0
```

As we can see on the output above there are no duplicate rows and no missing values.Also we have only one dataframe which is telecom_dataset and we dont need to do any combination.

Analyzing the telecom dataset using NumPy and Pandas libraries.

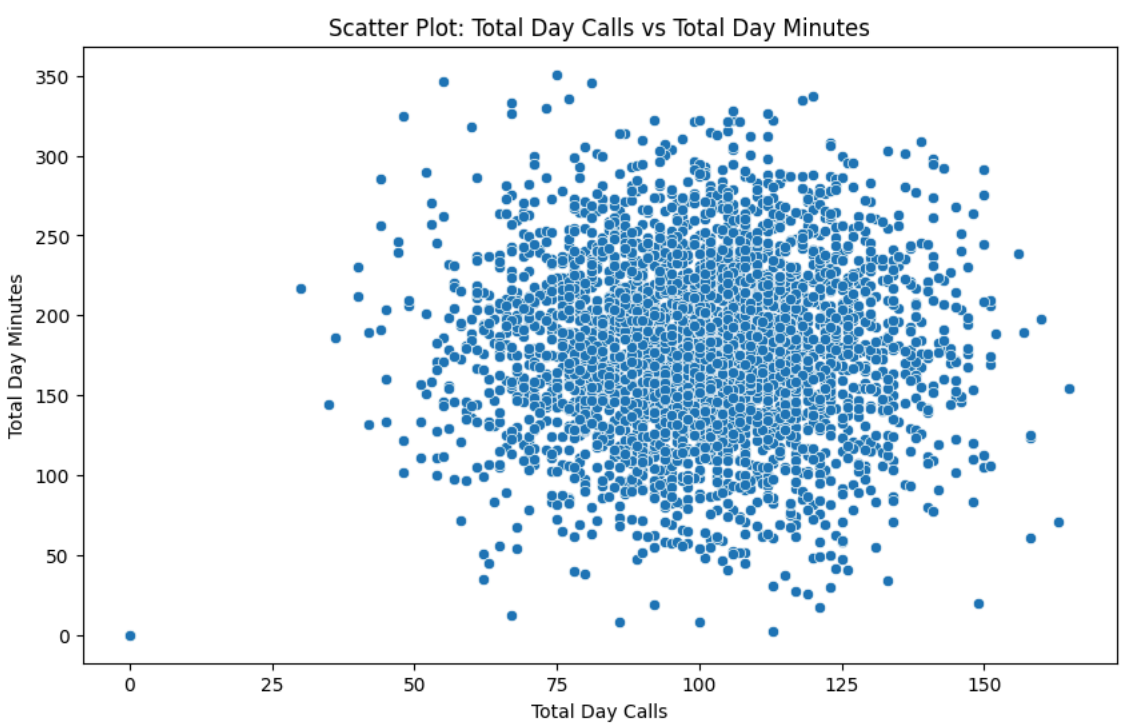
```
# Calculate commonly used statistics using Pandas
statistics_pandas = df.describe()
print("Statistics using Pandas:\n", statistics_pandas)

# Calculate statistics using NumPy for a specific columns, 'total day calls','total intl charge'
mean_total_day_calls = np.mean(df['total day calls'])
median_total_day_calls = np.median(df['total day calls'])
std_total_day_calls = np.std(df['total day calls'])
min_total_day_calls = np.min(df['total day calls'])
max_total_intl_charge = np.max(df['total intl charge'])
percentiles_total_day_calls = np.percentile(df['total day calls'], [25, 50, 75])

# Print the calculated statistics for 'total day calls','total intl charge'
print("\nMean of 'total day calls':", mean_total_day_calls)
print("Median of 'total day calls':", median_total_day_calls)
print("Standard Deviation of 'total day calls':", std_total_day_calls)
print("Minimum of 'total day calls':", min_total_day_calls)
print("Maximum of 'total intl charge':", max_total_intl_charge)
print("25th, 50th, and 75th percentiles of 'total day calls':", percentiles_total_day_calls)
```

```
Statistics using Pandas:
        account length    area code  number vmail messages  total day minutes  \
count     3333.000000  3333.000000            3333.000000        3333.000000
mean       101.064806   437.182418               8.099010         179.775098
std         39.822106    42.371290              13.688365          54.467389
min          1.000000   408.000000               0.000000           0.000000
25%         74.000000   408.000000               0.000000         143.700000
50%        101.000000   415.000000               0.000000         179.400000
75%        127.000000   510.000000              20.000000         216.400000
max        243.000000   510.000000              51.000000         350.800000

       total day calls  Predicted total day calls  total day charge  \
count      3333.000000                3333.000000       3333.000000
mean        100.435644                 100.374737         30.562307
std          20.069084                  20.141806          9.259435
min           0.000000                   0.000000          0.000000
25%          87.000000                  87.000000         24.430000
50%         101.000000                 101.000000         30.500000
75%         114.000000                 114.000000         36.790000
max         165.000000                 165.000000         59.640000

       total eve minutes  total eve calls  total eve charge  \
```

```
       total day calls  total day minutes  total day charge  \
count      3333.000000        3333.000000       3333.000000
mean        200.980348         100.114311         17.083540
std          50.713844          19.922625          4.310668
min           0.000000           0.000000          0.000000
25%         166.600000          87.000000         14.160000
50%         201.400000         100.000000         17.120000
75%         235.300000         114.000000         20.000000
max         363.700000         170.000000         30.910000

       total night minutes  total night calls  total night charge  \
count          3333.000000        3333.000000         3333.000000
mean            200.872037         100.107711            9.039325
std              50.573847          19.568609            2.275873
min              23.200000          33.000000            1.040000
25%             167.000000          87.000000            7.520000
50%             201.200000         100.000000            9.050000
75%             235.300000         113.000000           10.590000
max             395.000000         175.000000           17.770000

       total intl minutes  total intl calls  total intl charge  \
count         3333.000000       3333.000000        3333.000000
mean            10.237294          4.479448           2.764581
std              2.791840          2.461214           0.753773
min              0.000000          0.000000           0.000000
25%              8.500000          3.000000           2.300000
50%             10.300000          4.000000           2.780000
75%             12.100000          6.000000           3.270000
max             20.000000         20.000000           5.400000

       customer service calls
count             3333.000000
mean                 1.562856
std                  1.315491
min                  0.000000
25%                  1.000000
50%                  1.000000
```

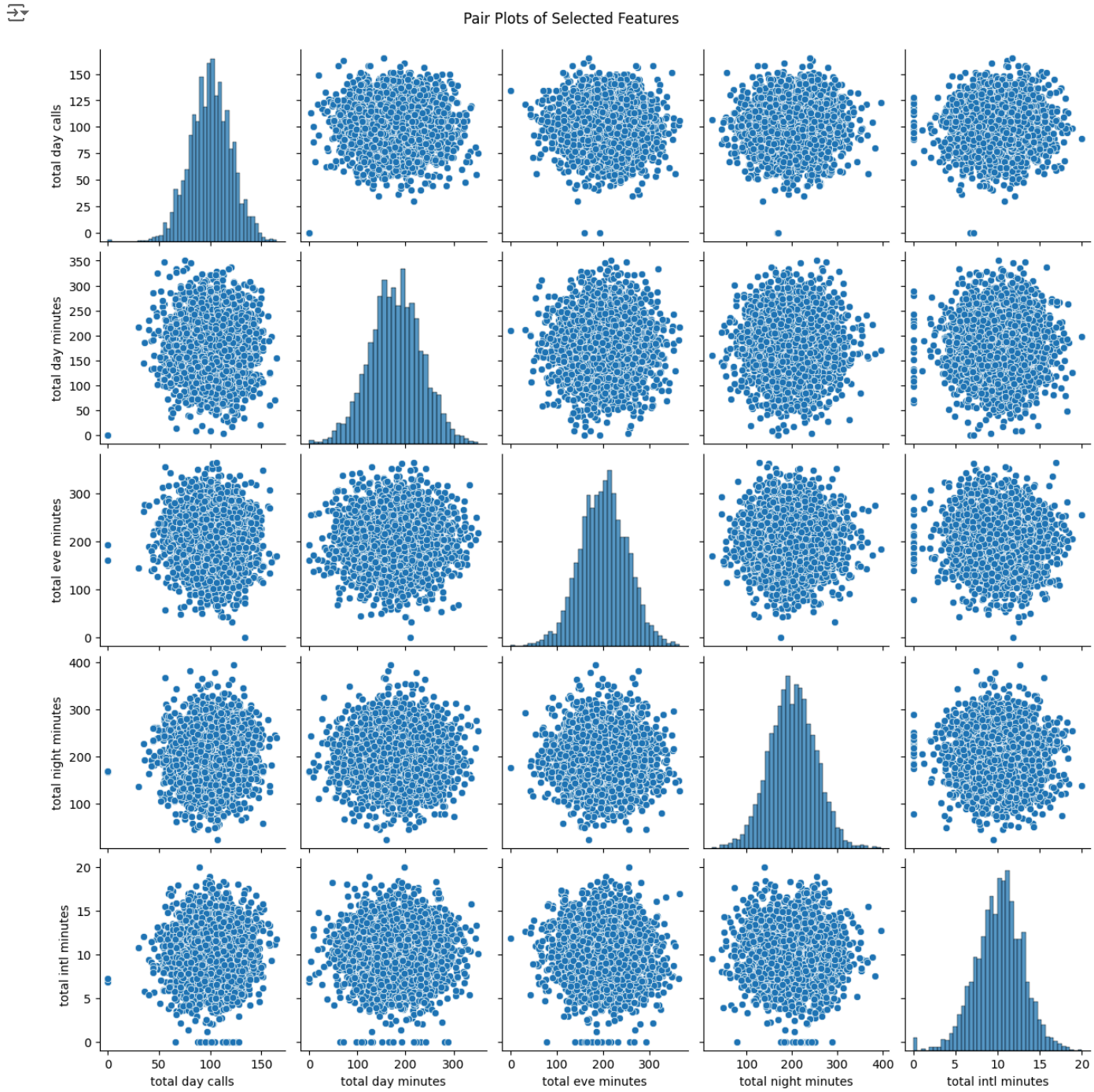
How do you create some plotsfor each pair of numerical features in the input dataframe? Provide examples of data visualization techniques commonly used in exploratory data analysis and demonstrate how to create them using libraries like Matplotlib or Seaborn. How would you decide which type of plot is most appropriate for visualizing the relationship between two numerical features? Could you demonstrate how to create pairwise relationship plots for numerical features in a dataset, such as scatter plots, pair plots, and correlation matrices, using Python libraries?

```
import matplotlib.pyplot as plt
import seaborn as sns

# Scatter plot between 'total day calls' and 'total day minutes'
plt.figure(figsize=(10, 6))
sns.scatterplot(x='total day calls', y='total day minutes', data=df)
plt.title('Scatter Plot: Total Day Calls vs Total Day Minutes')
plt.xlabel('Total Day Calls')
plt.ylabel('Total Day Minutes')
plt.show()
```



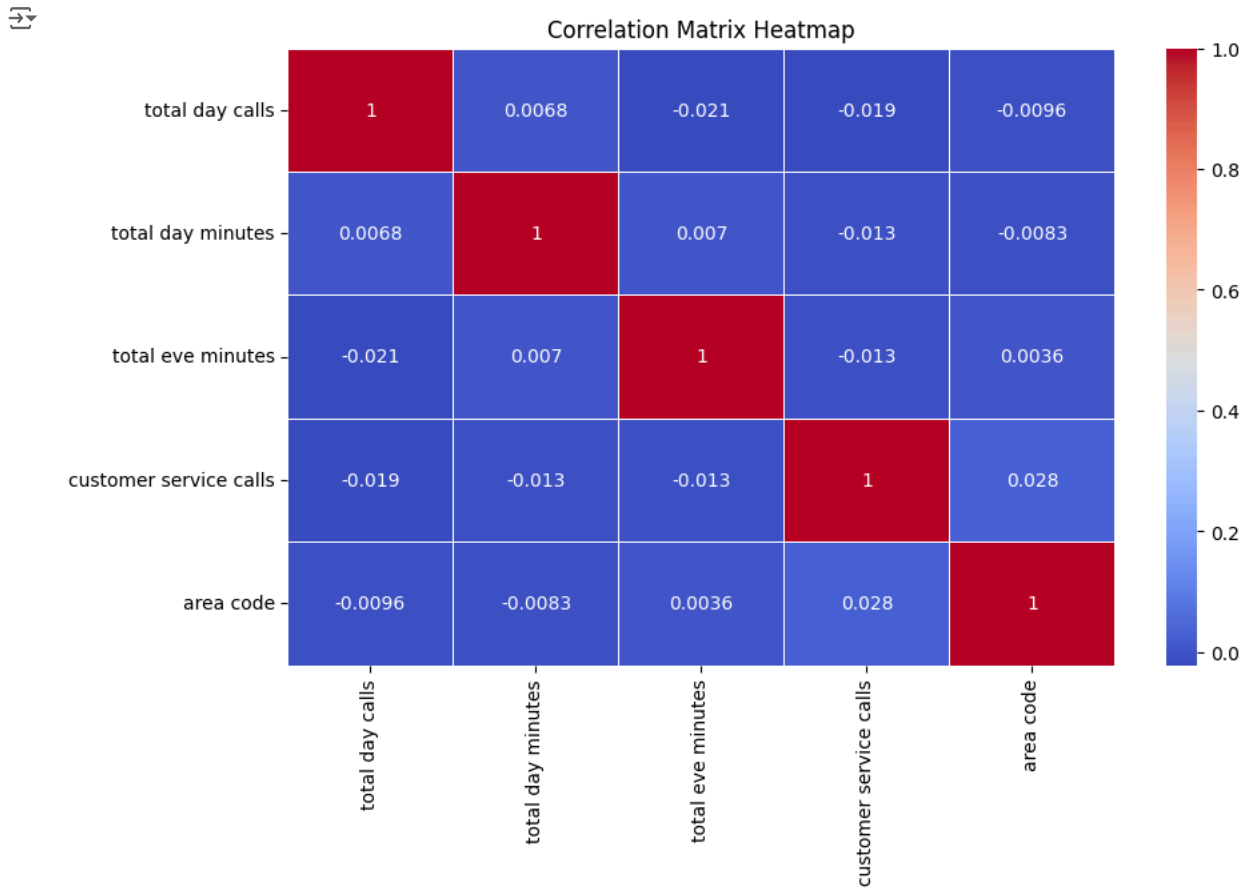
The scatter plot indicates that while there is some variability in the number of calls and total minutes, there is no clear linear relationship between these two variables. This suggests that factors other than just the number of calls influence the total duration of calls made by customers.

```
# Pair plot for a subset of numerical features
sns.pairplot(df[['total day calls', 'total day minutes', 'total eve minutes', 'total night minutes', 'total intl minutes']])
plt.suptitle('Pair Plots of Selected Features', y=1.02)
plt.show()
```



The pair plot provides a comprehensive view of the relationships and distributions of the selected numerical features in the dataset.

```
# Correlation matrix
correlation_matrix = df[['total day calls', 'total day minutes', 'total eve minutes', 'customer service calls', 'area code']].corr()

# Heatmap for correlation matrix
plt.figure(figsize=(10, 6))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', linewidths=0.5)
plt.title('Correlation Matrix Heatmap')
plt.show()
```



As shown in the correlation matrix there is no close relation to any of the features respectively.

Deliverables Section 3: Write the following queries and takes a screenshot ofthe output for each query. Implementation on MongoDB Atlas How to do implementation on MongoDB atlas? Provide two examples of inserting data into a MongoDB Atlas database. Designing NoSQL Schema for Company How to design a NoSQL schema for a company? Provide examples of a NoSQL schema designed for a company. Data Manipulation in MongoDB How to find, insert, delete, retrieve, and update data from a MongoDB database? Provide five examples of finding, inserting, deleting, retrieving, and updating data in MongoDB. Creating Plotsfor Numerical Features How do you create plots for each pair of numerical features in a dataframe? Provide examples of creating three different plots for pairwise numerical features

```
from pymongo import MongoClient
import dns
from urllib.parse import quote_plus
username = quote_plus('rcara10')
password = quote_plus('Eniiza2021@')

connection_string =f"mongodb+srv://{username}:{password}@cluster0.vm8mab5.mongodb.net/?retryWrites=true&w=majority&appName=Cluster0"
client = MongoClient(connection_string)
# Replacing with my actual database name
db = client.Section3DatabasesAssignment
collection = db.Costumers
```

```
from pymongo import MongoClient

db = client.Section3DatabasesAssignment
products_collection = db.products

# Single document to insert
product_data = {
    "Product_name": "Organic Apples",
    "Short_desc": "Fresh organic apples from local farms",
    "Dimensions": {"Length": 10, "Width": 10, "Height": 10},
    "Quantity_per_unit": "1 kg",
    "Avg_ratings": 4.6,
    "Std_price": 3.99,
    "Supp_price": 2.50,
    "Category": "fruits & vegetables",
    "Fresh": {
        "Category": "fruits & vegetables",
        "Best_before": 30,
        "Country_of_origin": "USA"
    }
}


result = products_collection.insert_one(product_data)
print(f"Inserted document id: {result.inserted_id}")
```

```
Inserted document id: 664cab058cfad3b888ba3614
```

```
from pymongo import MongoClient

db = client.Section3DatabasesAssignment
customers_collection = db.Costumers
customer_data = {
    "Name": "John Doe",
    "Gender": "M",
    "Age": 30,
    "Phone number": "1234567890",
    "Addresses": [{
        "House": "123",
        "Street": "Main St",
        "City": "Anytown",
        "Post code": "12345",
        "Location": {
            "Coordinates": [-122.4194, 37.7749]
        },
        "Current orders": [{
            "Date": "2024-05-21",
            "Order status": "Dispatched",
            "Order details": {
                "Total cost": 199.99,
                "partner_id": "partner123",
                "Shipping_id": "shipping123",
                "Supplier_id": "supplier123"
            },
            "Recommended products": [{
                "Product_id": "product123",
                "Avg_rating": 4.5
            }]
        }]
    }]
}

result = customers_collection.insert_one(customer_data)
print(f"Inserted customer document id: {result.inserted_id}")
```

```
Inserted customer document id: 664cb2ec8cfad3b888ba361e
```

Provide examples of a NoSQL schema designed for a company.

```
from pymongo import MongoClient
db = client.Section3DatabasesAssignment
# List collections
collections = db.list_collection_names()
print("Collections in database:", collections)
# Function to print documents from a collection
def print_documents(collection_name, limit=5):
    collection = db[collection_name]
    documents = collection.find().limit(limit)
    print(f"\nDocuments in {collection_name}:")
    for doc in documents:
        print(doc)

# Print documents from each collection
for collection_name in collections:
    print_documents(collection_name)
```

```
Collections in database: ['products', 'partnerHistory', 'dailyInventoryRecord', 'pastOrders', 'ratings', 'partners', 'Costumers',

Documents in products:
{'_id': 'CD2', 'name': 'Led Zepellin IV', 'short_desc': 'The informal setting at Headley Grange inspired the band, allowing them
{'_id': 'CD3', 'name': '21', 'short_desc': "Composed in the aftermath of the singer's separation from her then partner, the album
{'_id': 'CD4', 'name': 'The Wall', 'short_desc': 'It is a rock opera that explores Pink, a jaded rock star whose eventual self-im
{'_id': 'CD5', 'name': 'Back in Black', 'short_desc': 'The seventh studio album by Australian rock band AC/DC. It was released on
{'_id': 'CD6', 'name': 'Abbey Road', 'short_desc': 'Abbey Road incorporates styles such as rock, pop, blues, singer-songwriter, a

Documents in partnerHistory:
{'_id': ObjectId('63b852b6a7fc6363d52e5cbe'), 'partner_id': 'PA1', 'start_date': datetime.datetime(2023, 1, 2, 0, 0), 'end_date':
{'_id': ObjectId('63b854d4a7fc6363d52e5cc0'), 'partner_id': 'PA2', 'start_date': datetime.datetime(2023, 1, 2, 0, 0), 'end_date':
{'_id': ObjectId('63b8558ba7fc6363d52e5cc2'), 'partner_id': 'PA3', 'start_date': datetime.datetime(2023, 1, 2, 0, 0), 'end_date':
{'_id': ObjectId('63b855fba7fc6363d52e5cc4'), 'partner_id': 'PA3', 'start_date': datetime.datetime(2023, 1, 2, 0, 0), 'end_date':
{'_id': ObjectId('63b85681a7fc6363d52e5cc6'), 'partner_id': 'PA3', 'start_date': datetime.datetime(2023, 1, 2, 0, 0), 'end_date':

Documents in dailyInventoryRecord:
{'_id': {'supplier_id': 'W2', 'product_id': 'HA2', 'start_date': '02/01/2023 00:00', 'end_date': '02/01/2023 23:59'}, 'supplier_l
{'_id': {'supplier_id': 'ST4', 'product_id': 'FP11', 'start_date': '03/01/2023 00:00', 'end_date': '03/01/2023 23:59'}, 'supplier
{'_id': {'supplier_id': 'ST2', 'product_id': 'FP10', 'start_date': '03/01/2023 00:00', 'end_date': '03/01/2023 23:59'}, 'supplier
{'_id': {'supplier_id': 'W2', 'product_id': 'HA5', 'start_date': '03/01/2023 00:00', 'end_date': '03/01/2023 23:59'}, 'supplier_l
{'_id': {'supplier_id': 'ST5', 'product_id': 'FP9', 'start_date': '02/01/2023 00:00', 'end_date': '02/01/2023 23:59'}, 'supplier_

Documents in pastOrders:
{'_id': '20221001050939C4', 'order_date': datetime.datetime(2022, 10, 1, 5, 9), 'customer_id': 'C4', 'order_details': [{'product_
{'_id': '20221001102713C19', 'order_date': datetime.datetime(2022, 10, 1, 10, 27), 'customer_id': 'C19', 'order_details': [{'prod
{'_id': '20221001214036C16', 'order_date': datetime.datetime(2022, 10, 1, 21, 40), 'customer_id': 'C16', 'order_details': [{'prod
{'_id': '20221001221901C7', 'order_date': datetime.datetime(2022, 10, 1, 22, 19), 'customer_id': 'C7', 'order_details': [{'produc
{'_id': '20221001224411C6', 'order_date': datetime.datetime(2022, 10, 1, 22, 44), 'customer_id': 'C6', 'order_details': [{'produc

Documents in ratings:
{'_id': 'RA1', 'order_date': datetime.datetime(2022, 10, 1, 22, 44, 11), 'published_date': datetime.datetime(2022, 10, 2, 22, 44,
{'_id': 'RA2', 'order_date': datetime.datetime(2022, 10, 4, 16, 52, 33), 'published_date': datetime.datetime(2022, 10, 5, 16, 52,
{'_id': 'RA3', 'order_date': datetime.datetime(2022, 10, 4, 20, 32, 4), 'published_date': datetime.datetime(2022, 10, 5, 20, 32,
{'_id': 'RA4', 'order_date': datetime.datetime(2022, 10, 5, 3, 13, 11), 'published_date': datetime.datetime(2022, 10, 6, 3, 13, 1
{'_id': 'RA5', 'order_date': datetime.datetime(2022, 10, 6, 19, 37, 37), 'published_date': datetime.datetime(2022, 10, 7, 19, 37,

Documents in partners:
{'_id': 'PA1', 'name': 'Mike Dean', 'age': 20, 'gender': 'M', 'phone': '07618259974', 'email': 'mike.dean@gmail.com', 'bank_accou
{'_id': 'PA2', 'name': 'Robert Chaniago', 'age': 23, 'gender': 'M', 'phone': '07412744098', 'email': 'robert.chan@gmail.com', 'ba
{'_id': 'PA3', 'name': 'Hashim Ridwan', 'age': 34, 'gender': 'M', 'phone': '07212327676', 'email': 'hashim.ridwan@gmail.com', 'ba
{'_id': 'PA4', 'name': 'Sebastian Kanu', 'age': 45, 'gender': 'M', 'phone': '07316371076', 'email': 'se.kanu@gmail.com', 'bank_ac
{'_id': 'PA5', 'name': 'Alan Smith', 'age': 29, 'gender': 'M', 'phone': '07518346320', 'email': 'alan.smith@gmail.com', 'bank_acc

Documents in Costumers:
{'_id': 'C1', 'Customer': 'Gunner Ferrell', 'Gender': 'M', 'Age': 51, 'phone_number': 443454155475, 'addresses': [{'_id': 'AD1',
{'_id': 'C2', 'Customer': 'Lillie Costa', 'Gender': 'F', 'Age': 30, 'phone_number': 447137031760, 'addresses': [{'_id': 'AD2', 'h
{'_id': 'C3', 'Customer': 'Raelynn Dodson', 'Gender': 'F', 'Age': 44, 'phone_number': 443482883256, 'addresses': [{'_id': 'AD3',
```

How to find, insert, delete, retrieve, and update data from a MongoDB database? Provide five examples of finding, inserting, deleting, retrieving, and updating data in MongoDB.

```
{'_id': 'W1', 'name': 'Amazon UK MAN1', 'address': 'Manchester Airport, 6 Sunbank Ln, Altrincham', 'city': 'Manchester', 'post_co
```

```
query = {"Gender": "M"}
customer = db.Costumers.find_one(query)
print("Found Customer by Gender:", customer)
```

```
Found Customer by Gender: {'_id': 'C1', 'Customer': 'Gunner Ferrell', 'Gender': 'M', 'Age': 51, 'phone_number': 443454155475, 'addr
```