```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import csv
```

a) The system will allow the user to retrieve data from a CSV file using the csv module and

fundamental python (control structure and file processing) to perform the following:

- Load the data from a CSV file into memory using the csv reader function. The path to the file will be specified by the user then use these loaded data to perform following tasks: a1. Retrieve the model name, manufacturer, weight, price, and price unit for the device(s) based on the oem_id.

```
# Function to load data from CSV file
def load_data(file_path):
    data = []
    with open(file_path, 'r', encoding='utf-8') as file:
        csv_reader = csv.DictReader(file)
        for row in csv_reader:
            data.append(row)
    return data

# Function to retrieve device information based on oem_id
def retrieve_by_oem_id(data, oem_id):
    result = []
    for record in data:
        if record.get('oem_id') == oem_id:
            result.append({
                'model_name': record.get('model_name'),
                'manufacturer': record.get('manufacturer'),
                'weight': record.get('weight'),
                'price': record.get('price'),
                'price_unit': record.get('price_unit')
            })
    return result

# Main function
def main():
    file_path = input("Enter the path to the CSV file: ")
    data = load_data(file_path)

    oem_id = input("Enter the oem_id to retrieve device information: ")

    # Retrieve device information based on oem_id
    result_a1 = retrieve_by_oem_id(data, oem_id)

    # Display the result
    if result_a1:
        print("\nDevice Information for oem_id", oem_id, ":", result_a1)
    else:
        print("\nNo device found with the specified oem_id.")

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

```
Enter the path to the CSV file: /content/device_features.csv
Enter the oem_id to retrieve device information: MC400

Device Information for oem_id MC400 : [{'model_name': None, 'manufacturer': 'Lenovo', 'weight': None, 'price': '4999', 'price_unit'
```

a2. Retrieve the brand, model name, RAM capacity, market regions, and the date when the information was added for device(s) associated with a specified code name.

```
data = pd.read_csv(r"/content/device_features.csv")
data
```

2	A135UZKAVZW	Samsung	Galaxy A13 2022 Standard Edition TD-L...	28-03-22	04-03-22	
3	PAYB0000JP	Motorola	Moto G53j 5G 2023 Dual SIM TD-LTE JP 128GB XT2...	16-06-23	01-05-23	
4	MC400	Motorola	Moto G53 5G 2022 Premium Edition TD-LTE LATAM ...	01-01-23	01-12-22	
...	...	...	...	...	...	
1266	ZS661KS-6A020EU	Asus	ROG Phone 3 5G Extreme Edition Global Dual SIM...	18-08-20	23-07-20	ASUSTel
1267	PAG50053ISGB	Lenovo	Legion Phone Duel 5G Premium Edition Global Du...	01-10-20	22-07-20	
1268	MC361	Motorola	Moto G 5G Plus 2020 Global Dual SIM TD-LTE 128...	19-07-20	01-07-20	Motc
1269	G986UZPAXAA	Samsung	SM-G986U1 Galaxy S20+ 5G BTS Edition TD-LTE US...	16-07-20	14-06-20	
1270	A516UZKAATT	Samsung	SM-A516U Galaxy A51 5G TD-LTE US / SM-A516A	21-08-20	01-08-20	

1271 rows × 48 columns

```
def retrieve_device_info_by_codename(data, codename):
    # Ensure data is a list of dictionaries
    if not isinstance(data, list) or not all(isinstance(device, dict) for device in data):
        raise ValueError("Input data must be a list of dictionaries.")

    return [
        {
            'brand': device.get('brand'),
            'model_name': device.get('model'),
            'ram_capacity': device.get('ram_capacity'),
            'market_regions': device.get('market_regions'),
            'info_added_date': device.get('info_added_date')
        }
        for device in data
        if device.get('codename') == codename
    ]



# Main function
def main():
    file_path = input("Enter the path to the CSV file: ")
    csv_data = load_data(file_path)

    codename = input("Enter the codename to retrieve device information: ")

    # Retrieve device information based on codename
    result_a2 = retrieve_device_info_by_codename(csv_data, codename)

    # Display the result
    if result_a2:
        print("\nDevice Information for codename", codename, ":", result_a2)
    else:
        print("\nNo device found with the specified codename.")

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

```
Enter the path to the CSV file: /content/device_features.csv
Enter the codename to retrieve device information: Samsung A135

Device Information for codename Samsung A135 : [{'brand': 'Samsung', 'model_name': 'SM-A135U Galaxy A13 2022 Standard Edition TD-LT
```

a3. Retrieve the oem_id, release date, announcement date, dimensions, and device category of the device(s) based on a specified RAM capacity.

```
def retrieve_device_info_by_ram_capacity(data, ram_capacity):
    result = []
    for device in data:
        if device['ram_capacity'] == ram_capacity:
            result.append({
                'oem_id': device['oem_id'],
                'released_date': device['released_date'],
                'announced_date': device['announced_date'],
                'dimensions': device['dimensions'],
                'device_category': device['device_category']
            })
    return result


# Main function
def main():
    file_path = input("Enter the path to the CSV file: ")
    csv_data = load_data(file_path)

    ram_capacity = input("Enter the RAM capacity to retrieve device information: ")

    # Retrieve device information based on RAM capacity
    result_a3 = retrieve_device_info_by_ram_capacity(csv_data, ram_capacity)

    # Display the result
    if result_a3:
        print(f"\nDevice Information for RAM capacity {ram_capacity}:", result_a3)
    else:
        print(f"\nNo device found with the specified RAM capacity.")

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

```
Enter the path to the CSV file: /content/device_features.csv
Enter the RAM capacity to retrieve device information: 6

Device Information for RAM capacity 6: [{'oem_id': 'MC400', 'released_date': '01-01-23', 'announced_date': '01-12-22', 'dimensions'
```

a4. Retrieve information from your chosen columns and apply a specific condition that relates to an individual device. Please select at least three columns and one condition that differs from previous requirements.

```
# Function to retrieve device information based on a specific condition
def retrieve_with_condition(data):
    result = []
    for device in data:
        # Condition: Display refresh rate greater than 90 Hz (converted to int)
        if int(device['display_refresh_rate']) > 90:
            result.append({
                'oem_id': device['oem_id'],
                'display_refresh_rate': int(device['display_refresh_rate']),
                'display_type': device['display_type'],
                'market_regions': device['market_regions']
            })
    return result


# Main function
def main():
    file_path = input("Enter the path to the CSV file: ")
    csv_data = load_data(file_path)

    # Retrieve device information based on a specific condition
    result_a4 = retrieve_with_condition(csv_data)

    # Display the result
    if result_a4:
        print("\nDevice Information for devices with display refresh rate > 90 Hz:", result_a4)
    else:
        print("\nNo devices found with the specified condition.")

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

```
Enter the path to the CSV file: /content/device_features.csv

Device Information for devices with display refresh rate > 90 Hz: [{'oem_id': 'PAYB0000JP', 'display_refresh_rate': 120, 'display_t
```

```
from prettytable import PrettyTable

# Function to retrieve device information based on a specific condition
def retrieve_with_condition(data):
    result = []
    for device in data:
        # Condition: Display refresh rate greater than 90 Hz (converted to int)
        if int(device['display_refresh_rate']) > 90:
            result.append({
                'oem_id': device['oem_id'],
                'display_refresh_rate': int(device['display_refresh_rate']),
                'display_type': device['display_type'],
                'market_regions': device['market_regions']
            })
    return result

# Function to display result in table format
def display_result_table(result):
    if result:
        table = PrettyTable()
        table.field_names = ["OEM ID", "Display Refresh Rate", "Display Type", "Market Regions"]

        for device in result:
            table.add_row([device['oem_id'], device['display_refresh_rate'], device['display_type'], device['market_regions']])

        print(table)
    else:
        print("\nNo devices found with the specified condition.")

# Main function
def main():
    file_path = input("Enter the path to the CSV file: ")
    csv_data = load_data(file_path)

    # Retrieve device information based on a specific condition
    result_a4 = retrieve_with_condition(csv_data)

    # Display the result in table format
    display_result_table(result_a4)

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

```
|    N9860ZKGTGY     |         120          |      AM-OLEDdisplay      |
|    N985FZK8WWA     |         120          |      AM-OLEDdisplay      | Africa,Asia,Australia,Central America,Eastern Europe,Eu
|    N986BZWHEUA     |         120          |      AM-OLEDdisplay      |                                 Africa,Asia,Australia,Eastern E
|    Asus I003DD     |         144          |      AM-OLEDdisplay      |
|  ZS661KS-6A006IN   |         144          |      AM-OLEDdisplay      |
|  ZS661KS-6A009IN   |         144          |      AM-OLEDdisplay      |
|    Asus I003DD     |         144          |      AM-OLEDdisplay      |
|    Asus I003DD     |         144          |      AM-OLEDdisplay      |
|  ZS661KS-1A002EU   |         144          |      AM-OLEDdisplay      |             Africa,Asia,Australia,Eastern Europe,Europ
|    Asus I003DD     |         144          |      AM-OLEDdisplay      |
|  ZS661KS-6A020EU   |         144          |      AM-OLEDdisplay      |             Africa,Asia,Australia,Eastern Europe,Europ
|    PAG50053ISGB    |         144          |      AM-OLEDdisplay      |                                    Africa,Asia,Eastern Europe
|    G986UZPAXAA     |         120          |      AM-OLEDdisplay      |
+--------------------+----------------------+--------------------------+----------------------------------------------------------
```

The system will allow the user to analyse/query data using the pandas module to perform the

following:

- Load data from a CSV file into memory using the pandas module. Use the file path received from task a) for this purpose. After loading the data, proceed with the following tasks.

```
# Function to load data into a DataFrame using pandas
def load_data_pandas(file_path):
    return pd.read_csv(file_path)

# Function to retrieve device information based on oem_id using pandas
def retrieve_by_oem_id_pandas(data, oem_id):
    return data[data['oem_id'] == oem_id].loc[:, ['model', 'manufacturer', 'weight_gram', 'price', 'price_currency']]

# Function to retrieve device information based on code name using pandas
def retrieve_by_code_name_pandas(data, code_name):
    return data[data['codename'] == code_name].loc[:, ['brand', 'model', 'ram_capacity', 'market_regions', 'info_added_date']]

# Function to retrieve device information based on RAM capacity using pandas
def retrieve_by_ram_capacity_pandas(data, ram_capacity):
    return data[data['ram_capacity'] == ram_capacity].loc[:, ['oem_id', 'released_date', 'announced_date', 'dimensions', 'device_catego

# Function to retrieve device information based on a specific condition using pandas
def retrieve_with_condition_pandas(data):
    return data[data['display_refresh_rate'].astype(int) > 90].loc[:, ['oem_id', 'display_refresh_rate', 'display_type', 'market_region
```

```
from tabulate import tabulate
```

```
# Function to display result in table format
def display_result_table_pandas(result):
    if not result.empty:
        print(tabulate(result, headers='keys', tablefmt='pretty'))
    else:
        print("\nNo devices found with the specified condition.")


# Main function
def main():
    file_path = input("Enter the path to the CSV file: ")
    df_data = load_data_pandas(file_path)

    oem_id = input("Enter the oem_id to retrieve device information: ")
    result_a1 = retrieve_by_oem_id_pandas(df_data, oem_id)
    print("\nResult for Task a1:")
    display_result_table_pandas(result_a1)

    code_name = input("Enter the code name to retrieve device information: ")
    result_a2 = retrieve_by_code_name_pandas(df_data, code_name)
    print("\nResult for Task a2:")
    display_result_table_pandas(result_a2)

    ram_capacity = input("Enter the RAM capacity to retrieve device information: ")
    result_a3 = retrieve_by_ram_capacity_pandas(df_data, int(ram_capacity))
    print("\nResult for Task a3:")
    display_result_table_pandas(result_a3)

    result_a4 = retrieve_with_condition_pandas(df_data)
    print("\nResult for Task a4:")
    display_result_table_pandas(result_a4)

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

```
| 1113 |    A5260ZKAATT     |         120          |      AM-OLEDdisplay      |
| 1114 |    A5260ZKGCHC     |         120          |      AM-OLEDdisplay      |
| 1115 |    A5260ZKHCHC     |         120          |      AM-OLEDdisplay      |
| 1116 |    A526BZKHEUB     |         120          |      AM-OLEDdisplay      |                               Asia,Australia,Eas
| 1117 |    A526BZKDEUE     |         120          |      AM-OLEDdisplay      |                                          Central
| 1118 |    sweetin_pro     |         120          |      AM-OLEDdisplay      |
| 1119 |    sweetin_pro     |         120          |      AM-OLEDdisplay      |
| 1120 |    sweetin_pro     |         120          |      AM-OLEDdisplay      |
| 1121 |      sweetpro      |         120          |      AM-OLEDdisplay      |                                     Africa,Asia,
| 1126 |    haydnin_pro     |         120          |      AM-OLEDdisplay      |
| 1127 |     haydn_pro      |         120          |      AM-OLEDdisplay      |
| 1137 |    T570NZKEN20     |         120          | Color PLS TFT LCDdisplay | Africa,Asia,Australia,Central America,Eastern Eu
| 1138 |    T577UZKGN14     |         120          | Color PLS TFT LCDdisplay |
| 1150 |    G991BZVDEUA     |         120          |      AM-OLEDdisplay      |                     Africa,Asia,Australia,Easter
| 1151 |    G996BZVDEUA     |         120          |      AM-OLEDdisplay      |                     Africa,Asia,Australia,Easter
| 1152 |    G998BZSDEUA     |         120          |      AM-OLEDdisplay      |                     Africa,Asia,Australia,Easter
| 1201 |    W2021ZDDCHC     |         120          |      AM-OLEDdisplay      |
| 1210 |    F916UZAYXAA     |         120          |      AM-OLEDdisplay      |
| 1211 |    F9160ZNETGY     |         120          |      AM-OLEDdisplay      |
| 1212 |    F916UZKAUSC     |         120          |      AM-OLEDdisplay      |
| 1213 |    F916UZKASPR     |         120          |      AM-OLEDdisplay      |
| 1214 |    F916UZKATMB     |         120          |      AM-OLEDdisplay      |
| 1215 |    F916UZKAATT     |         120          |      AM-OLEDdisplay      |
| 1216 |    F916UZNAXAA     |         120          |      AM-OLEDdisplay      |
| 1217 |    F916UZKAVZW     |         120          |      AM-OLEDdisplay      |
| 1226 |    G781ULVMXAA     |         120          |      AM-OLEDdisplay      |
| 1227 |    G780FZBDEUE     |         120          |      AM-OLEDdisplay      |      Africa,Asia,Central America,Eastern Europe,
| 1228 |    G781UZBETMB     |         120          |      AM-OLEDdisplay      |
| 1229 |    G780FZGDEUE     |         120          |      AM-OLEDdisplay      |      Africa,Asia,Central America,Eastern Europe,
| 1230 |    G781BZBGMEA     |         120          |      AM-OLEDdisplay      | Africa,Asia,Australia,Central America,Eastern Eu
| 1232 |    N986NZNEKOD     |         120          |      AM-OLEDdisplay      |
| 1234 |    N986WZNAXAC     |         120          |      AM-OLEDdisplay      |
| 1236 |    N986UZKFVZW     |         120          |      AM-OLEDdisplay      |
| 1237 |    F916NZNAKOO     |         120          |      AM-OLEDdisplay      |
| 1245 |    F916BZKQMID     |         120          |      AM-OLEDdisplay      |                                   Asia,Australia
| 1246 |    N986WZKAXAC     |         120          |      AM-OLEDdisplay      |
| 1247 |    N986UZKAXAA     |         120          |      AM-OLEDdisplay      |
| 1248 |    N986UZKAUSC     |         120          |      AM-OLEDdisplay      |
| 1249 |    N986UZKASPR     |         120          |      AM-OLEDdisplay      |
| 1250 |    N986UZKAATT     |         120          |      AM-OLEDdisplay      |
| 1251 |    N986UZKAVZW     |         120          |      AM-OLEDdisplay      |
| 1252 |    N986UZKATMB     |         120          |      AM-OLEDdisplay      |
| 1253 |    N9860ZNGTGY     |         120          |      AM-OLEDdisplay      |
| 1254 |    N9860ZKGTGY     |         120          |      AM-OLEDdisplay      |
| 1255 |    N985FZK8WWA     |         120          |      AM-OLEDdisplay      | Africa,Asia,Australia,Central America,Eastern Eu
| 1258 |    N986BZWHEUA     |         120          |      AM-OLEDdisplay      |                         Africa,Asia,Australia,Ea
| 1259 |    Asus I003DD     |         144          |      AM-OLEDdisplay      |
| 1260 |  ZS661KS-6A006IN   |         144          |      AM-OLEDdisplay      |
| 1261 |  ZS661KS-6A009IN   |         144          |      AM-OLEDdisplay      |
| 1262 |    Asus I003DD     |         144          |      AM-OLEDdisplay      |
| 1263 |    Asus I003DD     |         144          |      AM-OLEDdisplay      |
| 1264 |  ZS661KS-1A002EU   |         144          |      AM-OLEDdisplay      |              Africa,Asia,Australia,Eastern Europ
| 1265 |    Asus I003DD     |         144          |      AM-OLEDdisplay      |
| 1266 |  ZS661KS-6A020EU   |         144          |      AM-OLEDdisplay      |              Africa,Asia,Australia,Eastern Europ
| 1267 |    PAG50053ISGB    |         144          |      AM-OLEDdisplay      |                              Africa,Asia,Eastern
| 1269 |    G986UZPAXAA     |         120          |      AM-OLEDdisplay      |
+------+--------------------+----------------------+--------------------------+-------------------------------------------------
```

b1. Identify the top 5 regions where a specific brand of devices was sold.

```
# Function to identify the top 5 regions for a specific brand
def top_regions_for_brand(data, brand):
    brand_data = data[data['brand'] == brand]
    top_regions = brand_data['market_regions'].str.split(', ').explode().value_counts().head(5)
    return top_regions
```

```
# Main function
def main():
    file_path = input("Enter the path to the CSV file: ")
    df_data = load_data_pandas(file_path)

    brand = input("Enter the brand to identify the top 5 regions: ")

    # Identify the top 5 regions for the specified brand
    top_regions = top_regions_for_brand(df_data, brand)

    # Display the result
    if not top_regions.empty:
        print(f"\nTop 5 regions for {brand} devices:")
        print(top_regions)
    else:
        print(f"\nNo data found for {brand} devices.")

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

```
Enter the path to the CSV file: /content/device_features.csv
Enter the brand to identify the top 5 regions: Motorola

Top 5 regions for Motorola devices:
Asia                                         21
North America                                21
North America,South America                  16
South America                                11
Asia,Eastern Europe,Europe,Western Europe     7
Name: market_regions, dtype: int64
```

b2. Analyse the average price of devices within a specific brand, all in the same currency.

```
# Function to analyze the average price of devices within a specific brand
def average_price_for_brand(data, brand):
    brand_data = data[data['brand'] == brand]
    average_price = brand_data.groupby('price_currency')['price'].mean()
    return average_price


# Main function
def main():
    file_path = input("Enter the path to the CSV file: ")
    df_data = load_data_pandas(file_path)

    brand = input("Enter the brand to analyze the average price: ")

    # Analyze the average price for the specified brand
    average_price = average_price_for_brand(df_data, brand)

    # Display the result
    if not average_price.empty:
        print(f"\nAverage prices for {brand} devices (all in the same currency):")
        print(average_price)
    else:
        print(f"\nNo data found for {brand} devices.")

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

```
Enter the path to the CSV file: /content/device_features.csv
Enter the brand to analyze the average price: Samsung

Average prices for Samsung devices (all in the same currency):
price_currency
AED    1.049000e+03
AUD    1.234000e+03
BRL    1.827571e+03
CAD    1.454945e+03
CNY    9.277125e+03
EUR    8.897697e+02
GBP    7.150000e+02
HKD    8.848100e+03
HUF    1.019467e+05
IDR    6.943444e+06
INR    8.221388e+04
JPY    1.467874e+05
KRW    1.202600e+06
MXN    9.010357e+03
MYR    1.974000e+03
PLN    1.724000e+03
RUB    2.190000e+04
SGD    5.980000e+02
```

```
TRY    6.999000e+03
TWD    2.142681e+04
USD    1.016129e+03
Name: price, dtype: float64
```

b3. Analyse the average mass for each manufacturer and display the list of average mass for all manufacturers.

```
# Function to analyze the average price of devices within a specific brand
def average_price_for_brand(data, brand):
    brand_data = data[data['brand'] == brand]
    average_price = brand_data.groupby('price_currency')['price'].mean()
    return average_price


# Main function
def main():
    file_path = input("Enter the path to the CSV file: ")
    df_data = load_data_pandas(file_path)

    brand = input("Enter the brand to analyze the average price: ")

    # Analyze the average price for the specified brand
    average_price = average_price_for_brand(df_data, brand)

    # Display the result
    if not average_price.empty:
        print(f"\nAverage prices for {brand} devices (all in the same currency):")
        print(average_price)
    else:
        print(f"\nNo data found for {brand} devices.")

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

```
Enter the brand to analyze the average price: Samsung

Average prices for Samsung devices (all in the same currency):
price_currency
AED    1.049000e+03
AUD    1.234000e+03
BRL    1.827571e+03
CAD    1.454945e+03
CNY    9.277125e+03
EUR    8.897697e+02
GBP    7.150000e+02
HKD    8.848100e+03
HUF    1.019467e+05
IDR    6.943444e+06
INR    8.221388e+04
JPY    1.467874e+05
KRW    1.202600e+06
MXN    9.010357e+03
MYR    1.974000e+03
PLN    1.724000e+03
RUB    2.190000e+04
SGD    5.980000e+02
TRY    6.999000e+03
TWD    2.142681e+04
USD    1.016129e+03
Name: price, dtype: float64
```

b4. Analyse the data to derive meaningful insights based on your unique selection, distinct from the previous requirements.

```
# Function to derive insights based on screen size and resolution
def derive_screen_insights(data):
    # Calculate pixel density (pixels per inch)
    data['pixel_density'] = (data['x_resolution']**2 + data['y_resolution']**2)**0.5 / data['display_diagonal']

    # Sort data by pixel density in descending order
    sorted_data = data.sort_values(by='pixel_density', ascending=False)

    # Identify device with the highest pixel density
    max_pixel_density_device = sorted_data.iloc[0]

    return max_pixel_density_device


# Main function
def main():
    file_path = input("Enter the path to the CSV file: ")
    df_data = load_data_pandas(file_path)
    # Derive insights based on screen size and resolution
    max_pixel_density_device = derive_screen_insights(df_data)

    # Display the result
    print("\nDevice with the highest pixel density:")
    print(max_pixel_density_device)

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

```
Enter the path to the CSV file: /content/device_features.csv

Device with the highest pixel density:
oem_id                                                                   XQBE62/B
brand                                                                        Sony
model                             Xperia Pro-I 2021 5G Dual SIM TD-LTE NA XQ-BE62
released_date                                                            11-12-21
announced_date                                                           26-10-21
hardware_designer                                                            Sony
manufacturer                                                                 Sony
codename                                                             Sony PDX-217
general_extras                       Haptic touch feedback,Tactile touch feedback
device_category                                                        Smartphone
width                                                                        72.0
height                                                                      166.0
depth                                                                         8.9
dimensions                                              2.83x6.54x0.35 inches
weight_gram                                                                 211.0
price                                                                     1799.99
price_currency                                                                USD
platform                                                                  Android
operating_system                                            Google Android 11 (R)
software_extras                 Voice Command,Navigation software,Augmented Re...
cpu_clock                                                                    2842
cpu                             Qualcomm Snapdragon 888 5G SM8350 (Lahaina), 2...
ram_type                                                             LPDDR5 SDRAM
ram_capacity                                                                   12
non_volatile_memory_capacity                                                512.0
display_hole                                                               1-hole
display_diagonal                                                            165.1
horizontal_full_bezel_width                                                  7.02
display_area_utilization                                                     82.5
pixel_density                                                           25.300548
display_type                                                       AM-OLEDdisplay
number_of_display_scales                                                     16.8
display_refresh_rate                                                          120
graphical_controller                                          Qualcomm Adreno 660
supported_cellular_bands        GSM850,GSM900,GSM1800,GSM1900,UMTS2100 (B1),UM...
sim_card_slot                                                      Nano-SIM (4FF)
usb                                            USB 3.0 / 3.1 Gen 1 / 3.2 Gen 1x1
usb_services                    USB charging,USB fast charging,USB Host,USB OT...
usb_connector                                                    USB C reversible
max_charging_power                                                           30.0
bluetooth                                                           Bluetooth 5.2
WLAN                            802.11a,802.11b,802.11g,802.11n,802.11ac,802.11ax
additional_sensors                                    FP sensor,L sensor,P sensor
battery_capacity                                                 4500 mAh battery
market_regions                                                      North America
info_added_date                                                    06-11-21 09:47
x_resolution                                                                 1644
y_resolution                                                                 3840
Name: 858, dtype: object
```

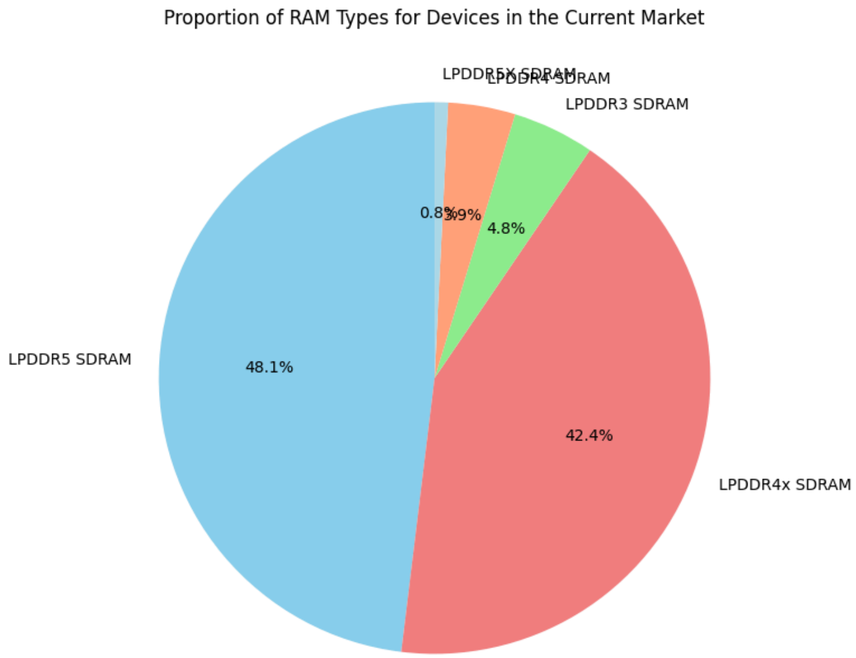
c) The system will allow the user to visualise the data using the matplotlib module as follows:

- Load data from a CSV file into memory. Use the file path received from task a) for this purpose. After loading the data, proceed with the following tasks. c1. Create a chart to visually represent the proportion of RAM types for devices in the current market.

```
# Function to create a chart for the proportion of RAM types with different colors
def create_ram_type_chart(data):
    ram_type_counts = data['ram_type'].value_counts()

    # Define different colors for each RAM type
    colors = ['skyblue', 'lightcoral', 'lightgreen', 'lightsalmon', 'lightblue']

    plt.figure(figsize=(8, 8))
    plt.pie(ram_type_counts, labels=ram_type_counts.index, autopct='%1.1f%%', startangle=90, colors=colors)
    plt.title('Proportion of RAM Types for Devices in the Current Market')
    plt.show()
```

```
# Main function
def main():
    file_path = input("Enter the path to the CSV file: ")
    df_data = load_data_pandas(file_path)

    # Create a chart for the proportion of RAM types
    create_ram_type_chart(df_data)

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

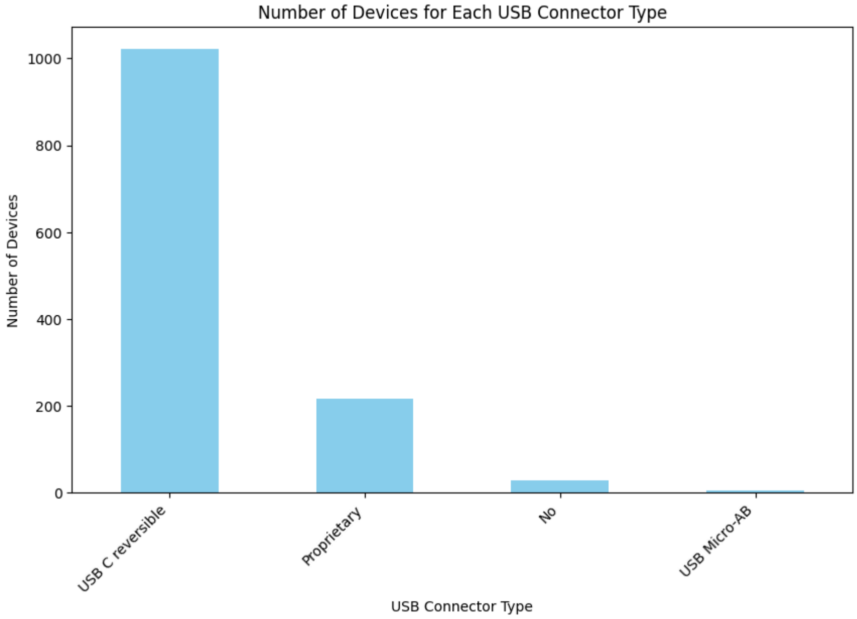
```
Enter the path to the CSV file: /content/device_features.csv
```



c2. Create a chart to visually compare the number of devices for each USB connector type

```
# Function to create a chart for the number of devices for each USB connector type
def create_usb_connector_chart(data):
    usb_connector_counts = data['usb_connector'].value_counts()

    plt.figure(figsize=(10, 6))
    usb_connector_counts.plot(kind='bar', color='skyblue')
    plt.title('Number of Devices for Each USB Connector Type')
    plt.xlabel('USB Connector Type')
    plt.ylabel('Number of Devices')
    plt.xticks(rotation=45, ha='right')
    plt.show()


# Main function
def main():
    file_path = input("Enter the path to the CSV file: ")
    df_data = load_data_pandas(file_path)

    # Create a chart for the number of devices for each USB connector type
    create_usb_connector_chart(df_data)

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

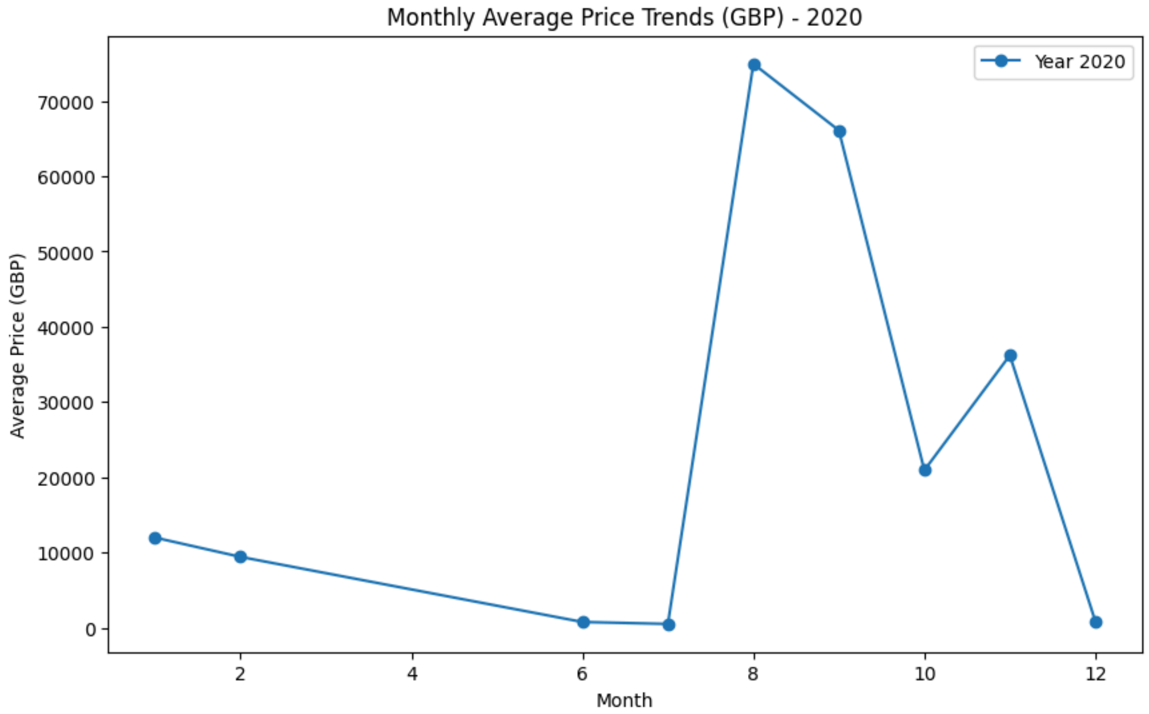
```
Enter the path to the CSV file: /content/device_features.csv
```



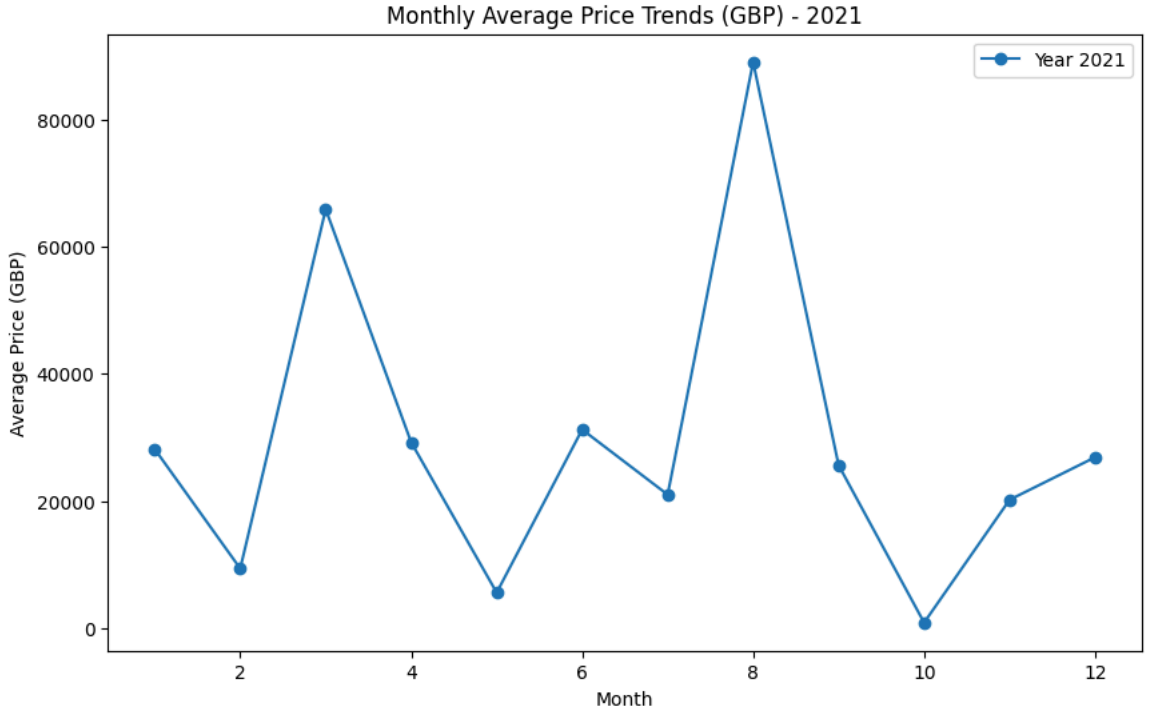
c3. Create separate charts illustrating the monthly average price trends (in GBP) for devices released in each year from 2020 to 2023. Each chart should focus on a specific year.

```
# Function to convert price to GBP if not already in GBP
def convert_to_gbp(row):
    if row['price_currency'] != 'GBP':
        # Assuming a conversion factor (adjust as needed)
        return row['price'] * 0.85
    return row['price']


# Function to create a chart for the monthly average price trends for each year
def create_price_trends_charts(data):
    # Convert prices to GBP if not already in GBP
    data['price_gbp'] = data.apply(convert_to_gbp, axis=1)

    # Extract the year from the 'released_date' column
    data['year'] = pd.to_datetime(data['released_date']).dt.year

    # Filter data for the years 2020 to 2023
    for year in range(2020, 2024):
        year_data = data[data['year'] == year]

        # Group data by month and calculate the average price for each month
        monthly_avg_price = year_data.groupby(pd.to_datetime(year_data['released_date']).dt.month)['price_gbp'].mean()

        plt.figure(figsize=(10, 6))
        plt.plot(monthly_avg_price.index, monthly_avg_price.values, marker='o', label=f'Year {year}')
        plt.title(f'Monthly Average Price Trends (GBP) - {year}')
        plt.xlabel('Month')
        plt.ylabel('Average Price (GBP)')
        plt.legend()
        plt.show()


# Main function
def main():
    file_path = input("Enter the path to the CSV file: ")
    df_data = load_data_pandas(file_path)

    # Create charts for the monthly average price trends for each year
    create_price_trends_charts(df_data)

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

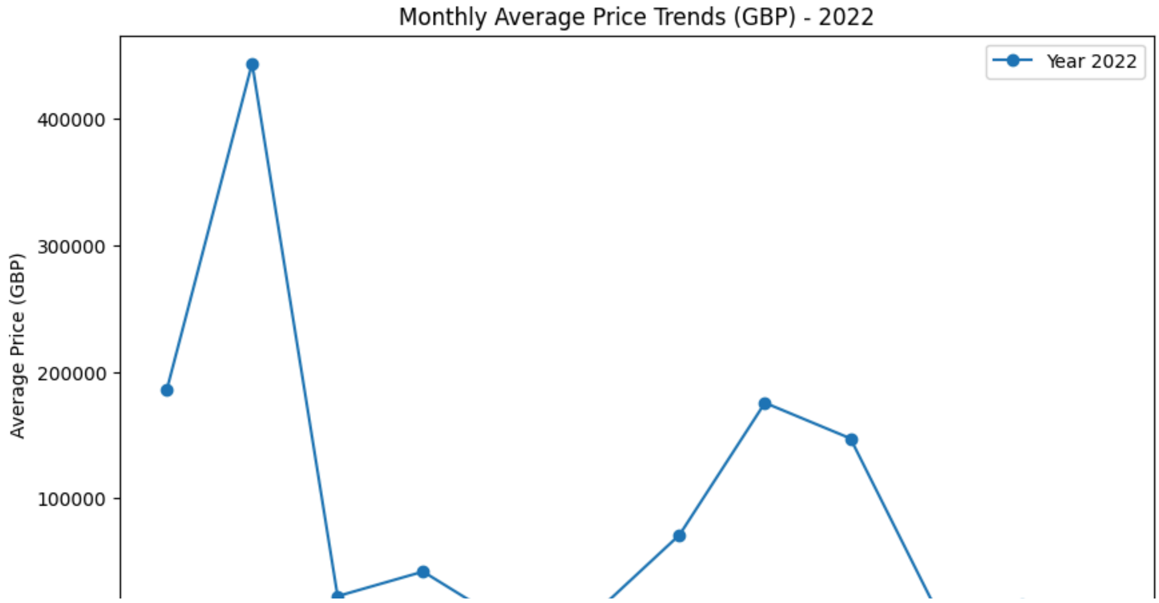
```
Enter the path to the CSV file: /content/device_features.csv
```



Monthly Average Price Trends (GBP) - 2020



Monthly Average Price Trends (GBP) - 2021



Monthly Average Price Trends (GBP) - 2022