Python Task 1

Question 1: Car Matrix Generation

Under the function named `generate car matrix` write a logic that takes the `dataset-1.csv` as a DataFrame. Return a new DataFrame that follows the following rules:

- values from `id_2` as columns
- values from `id_1` as index
- dataframe should have values from `car` column
- diagonal values should be 0.

id_2	801	802	803	804	805	806	807	808	809	821	\
id_1											
801	0.00	2.80	6.00	7.70	11.70	13.40	16.90	19.60	21.00	23.52	
802	2.80	0.00	3.40	5.20	9.20	10.90	14.30	17.10	18.50	20.92	
803	6.00	3.40	0.00	2.00	6.00	7.70	11.10	13.90	15.30	17.72	
804	7.70	5.20	2.00	0.00	4.40	6.10	9.50	12.30	13.70	16.12	
805	11.70	9.20	6.00	4.40	0.00	2.00	5.40	8.20	9.60	12.02	
806	13.40	10.90	7.70	6.10	2.00	0.00	3.80	6.60	8.00	10.42	
807	16.90	14.30	11.10	9.50	5.40	3.80	0.00	2.90	4.30	6.82	
808	19.60	17.10	13.90	12.30	8.20	6.60	2.90	0.00	1.70	4.12	
809	21.00	18.50	15.30	13.70	9.60	8.00	4.30	1.70	0.00	2.92	
821	23.52	20.92	17.72	16.12	12.02	10.42	6.82	4.12	2.92	0.00	
822	24.67	22.07	18.87	17.27	13.17	11.57	7.97	5.27	4.07	1.80	
823	26.53	23.93	20.73	19.13	15.03	13.43	9.83	7.13	5.93	3.67	
824	27.92	25.32	22.12	20.52	16.42	7.80	11.22	8.52	7.32	5.06	
825	29.08	26.48	23.28	21.68	17.58	15.98	12.38	9.68	8.48	6.22	
826	30.87	28.27	25.07	23.47	19.37	17.77	14.17	11.47	10.27	8.01	
827	32.53	29.93	26.73	25.13	21.03	19.43	15.83	13.13	11.93	9.43	
829	36.32	33.72	30.52	28.92	24.82	23.22	19.62	16.92	15.72	13.26	
830	38.27	35.67	32.47	30.87	26.77	25.17	21.57	18.87	17.67	15.17	
831	39.24	36.64	33.44	31.84	27.74	26.14	22.54	19.84	18.64	16.15	

Question 2: Car Type Count Calculation

Create a Python function named `get_type_count` that takes the `dataset-1.csv` as a DataFrame. Add a new categorical column `car_type` based on values of the column `car`:

- `low` for values less than or equal to 15,
- `medium` for values greater than 15 and less than or equal to 25,
- `high` for values greater than 25.

Calculate the count of occurrences for each `car_type` category and return the result as a dictionary. Sort the dictionary alphabetically based on keys.

```
{'high': 56, 'low': 196, 'medium': 89}
```

Question 3: Bus Count Index Retrieval

Create a Python function named `get_bus_indexes` that takes the `dataset-1.csv` as a DataFrame. The function should identify and return the indices as a list (sorted in ascending order) where the `bus` values are greater than twice the mean value of the `bus` column in the DataFrame.

```
[2, 7, 12, 17, 25, 30, 54, 64, 70, 97, 144, 145, 149, 154, 160, 201, 206, 210, 215, 234, 235, 245, 250, 309, 314, 319, 322, 32
3, 334, 340]
```

Question 4: Route Filtering

Create a python function `filter_routes` that takes the `dataset-1.csv` as a DataFrame. The function should return the sorted list of values of column `route` for which the average of values of `truck` column is greater than 7.

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

Question 5: Matrix Value Modification

Create a Python function named `multiply_matrix` that takes the resulting DataFrame from Question 1, as input and modifies each value according to the following logic:

- If a value in the DataFrame is greater than 20, multiply those values by 0.75,
- If a value is 20 or less, multiply those values by 1.25.

The function should return the modified DataFrame which has values rounded to 1 decimal place.

	A	B	C
0	0.0	18.8	15.0
1	23.4	22.5	22.5
2	12.5	6.2	20.6

Question 6: Time Check

You are given a dataset, `dataset-2.csv`, containing columns `id`, `id_2`, and timestamp (`startDay`, `startTime`, `endDay`, `endTime`). The goal is to verify the completeness of the time data by checking whether the timestamps for each unique (`id`, `id_2`) pair cover a full 24-hour period (from 12:00:00 AM to 11:59:59 PM) and span all 7 days of the week (from Monday to Sunday).
Create a function that accepts `dataset-2.csv` as a DataFrame and returns a boolean series that indicates if each (`id`, `id_2`) pair has incorrect timestamps. The boolean series must have multi-index (`id`, `id_2`).

```
Index(['id', 'name', 'id_2', 'startDay', 'startTime', 'endDay', 'endTime',
       'able2Hov2', 'able2Hov3', 'able3Hov2', 'able3Hov3', 'able5Hov2',
       'able5Hov3', 'able4Hov2', 'able4Hov3'],
      dtype='object')
id       id_2
1014000  -1          False
1014002  -1          False
1014003  -1          False
1030000  -1          False
          1030002    False
                     ...
1330016   1330006    False
          1330008    False
          1330010    False
          1330012    False
          1330014    False
Length: 9254, dtype: bool
```

Python Task 2

Question 1: Distance Matrix Calculation

Create a function named `calculate_distance_matrix` that takes the `dataset-3.csv` as input and generates a DataFrame representing distances between IDs.
The resulting DataFrame should have cumulative distances along known routes, with diagonal values set to 0. If distances between toll locations A to B and B to C are known, then the distance from A to C should be the sum of these distances. Ensure the matrix is symmetric, accounting for bidirectional distances between toll locations (i.e. A to B is equal to B to A).

	1001400.0	1001402.0	1001404.0	1001406.0	1001408.0	1001410.0	\
1001400.0	0.0	9.7	29.9	45.9	67.6	78.7	
1001402.0	inf	0.0	20.2	36.2	57.9	69.0	
1001404.0	inf	inf	0.0	16.0	37.7	48.8	
1001406.0	inf	inf	inf	0.0	21.7	32.8	
1001408.0	inf	inf	inf	inf	0.0	11.1	
1001410.0	inf	inf	inf	inf	inf	0.0	
1001412.0	inf	inf	inf	inf	inf	inf	
1001414.0	inf	inf	inf	inf	inf	inf	
1001416.0	inf	inf	inf	inf	inf	inf	
1001418.0	inf	inf	inf	inf	inf	inf	
1001420.0	inf	inf	inf	inf	inf	inf	
1001422.0	inf	inf	inf	inf	inf	inf	
1001424.0	inf	inf	inf	inf	inf	inf	
1001426.0	inf	inf	inf	inf	inf	inf	
1001428.0	inf	inf	inf	inf	inf	inf	
1001430.0	inf	inf	inf	inf	inf	inf	
1001432.0	inf	inf	inf	inf	inf	inf	
1001434.0	inf	inf	inf	inf	inf	inf	

Question 2: Unroll Distance Matrix

Create a function `unroll_distance_matrix` that takes the DataFrame created in Question 1. The resulting DataFrame should have three columns: columns `id_start`, `id_end`, and `distance`.
All the combinations except for same `id_start` to `id_end` must be present in the rows with their distance values from the input DataFrame.

	id_start	id_end	distance
0	1001400.0	1001402.0	9.7
1	1001400.0	1001404.0	29.9
2	1001400.0	1001406.0	45.9
3	1001400.0	1001408.0	67.6
4	1001400.0	1001410.0	78.7
...	...	...	...
1801	1001472.0	1001462.0	inf
1802	1001472.0	1001464.0	inf
1803	1001472.0	1001466.0	inf
1804	1001472.0	1001468.0	inf
1805	1001472.0	1001470.0	inf

[1806 rows x 3 columns]

Question 3: Finding IDs within Percentage Threshold

Create a function `find_ids_within_ten_percentage_threshold` that takes the DataFrame created in Question 2 and a reference value from the `id_start` column as an integer. Calculate average distance for the reference value given as an input and return a sorted list of values from `id_start` column which lie within 10% (including ceiling and floor) of the reference value's average.

```
[]
```

Question 4: Calculate Toll Rate

Create a function `calculate_toll_rate` that takes the DataFrame created in Question 2 as input and calculates toll rates based on vehicle types.
The resulting DataFrame should add 5 columns to the input DataFrame: `moto`, `car`, `rv`, `bus`, and `truck` with their respective rate coefficients. The toll rates should be calculated by multiplying the distance with the given rate coefficients for each vehicle type:

- 0.8 for `moto`
- 1.2 for `car`
- 1.5 for `rv`
- 2.2 for `bus`
- 3.6 for `truck`

	id_start	id_end	distance	moto	car	rv	bus	truck
0	1001400.0	1001402.0	9.7	7.76	11.64	14.55	21.34	34.92
1	1001400.0	1001404.0	29.9	23.92	35.88	44.85	65.78	107.64
2	1001400.0	1001406.0	45.9	36.72	55.08	68.85	100.98	165.24
3	1001400.0	1001408.0	67.6	54.08	81.12	101.40	148.72	243.36
4	1001400.0	1001410.0	78.7	62.96	94.44	118.05	173.14	283.32
...	...	...	...	...	...	...	...	...
1801	1001472.0	1001462.0	inf	inf	inf	inf	inf	inf
1802	1001472.0	1001464.0	inf	inf	inf	inf	inf	inf
1803	1001472.0	1001466.0	inf	inf	inf	inf	inf	inf
1804	1001472.0	1001468.0	inf	inf	inf	inf	inf	inf
1805	1001472.0	1001470.0	inf	inf	inf	inf	inf	inf

[1806 rows x 8 columns]