Technical Interview

1. The Data

Imagine that you work for a car seller.

Your director just found a dataset lost in your legacy systems and he sends you a few links related to it:

- Dataset page - This is a UCI open dataset.

He has no idea what this data is about and he wants you to find out.

You cannot use "ucimlrepo" to import the data. You should download the files and import them manually.

1.1. Handling the Data

Your task is to analyze the data and report him your findings!

No worries. Your lead just backed you up and sent you a list of what is expected from:

- Create a .py module that processes and cleans the data
- What insights can you take from the data?
- What data quality issues did you find?

You are free to use any library and any frameworks of your choice.

However, doing it in Python is highly valuable!

1.2. Additional Challenges

1.2.1.

How would you create a database with the previous dataset?

1.2.2.

Imagine that you would be receiving new datasets like this one, in a regular basis.
How would you handle it so that you do not have to run the code manually everytime you "download" a new chunk of data?
This exercise is totally free, as much as you clarify the assumptions taken in your solution.

2. Algotithmic Thinking

2.1.

Please write a function that loops over a list of numbers and prints the **current number** and **the average** of the printed numbers (including the current number).
Run it for the following list of numbers:

``` 0.5, 3.0, 7.5, 14.0, 22.5, 33.0, 45.5, 60.0, 76.5, 95.0, 115.5, 138.0, 162.5, 189.0, 217.5, 248.0, 280.5, 315.0, 351.5 ```

In [ ]:

```
# ls_test = [0.5, 3.0, 7.5, 14.0, 22.5, 33.0, 45.5, 60.0, 76.5, 95.0, 115.5, 138.0,
```

2.2.

Imagine that you are using an API that can only receive words inside brackets, and the only valid brackets are `'{'`, `'}'`, `'['` and `']'`.
The API can accept a string with more than one type of brackets, and it is only valid if the brackets are closed in the correct order (e.g., these are valid strings: `'{[Hello World]}'`, `'[[Hello World]]'`, `'{Hello World}'`).
Please, write a function to determine if a given string is valid.
If a string is not valid, it shall print an adequate error message.

In [ ]:

2.3.

Given a collection of intervals (e.g., `([2, 4], [2, 5], [3, 6], [9, 11])`), write a function that merges all overlapping intervals.

For example, ``` input: ([2, 4], [2, 5], [3, 6], [9, 11]) -> ouput: ([2, 6], [9, 11]) ``` .

In [ ]:

2.4.

Please write a function that returns the median of two or more arrays.
The function shall be able to receive as may arrays as we want (`f(array_1, array_2, ..., array_n)`)

For example, ``` input: ([4, 2, 1], [2, 5], [7, 6]) -> ouput: 4 ``` .

In [ ]:

3. SQL

You have 3 tables in your company's database:

- tb_employee

  Employees' information

  Columns:

  - employee_id - Primary Key
  - age
  - gender
  - position
  - department
  - country
  - district

- tb_salary:

  It contains all salary payments to a given employee. Each payment

  Columns:

  - payment_id - Primary Key
  - month_id (format: YYYYmm, e.g., 200202)
  - employee_id - Foreign Key (non-unique)
  - salary_value

- tb_reference_salary:

  It contains the salary bands for each position

  Columns:

  - position - primary key (e.g., controller)
  - year - (from 2002 to 2023)
  - maximum_ref_value
  - minimum_ref_value

Your challange is to write the queries above.

3.1.

Please provide a query that returns a list of employees **in your department ("market_insights")** from the **"Plzen"** district in the Czech Republic, sorted by those who have a current salary income **above the maximum reference value**.

Bonus: could you please arrange the list in order, prioritizing those who have been in such a situation for a longer period of time?

In [ ]:

3.2.

Please provide a query that returns a list of employees **in the "supercars" department** from the **Woking Borough** district in the county of Surrey (England), sorted by those who had the lower anual salary income in 2010.

```
In [ ]:
```