# Middle East Investment Bank Technical Task Solution
Hossein Mohseni

## Pipeline Design and Architecture

The ETL (Extract, Transform, Load) pipeline implemented in the provided code follows a simple design to fetch data from the fipiran API, transform it into a structured format, and load it into separate tables of our relational database.

Here is an overview of the pipeline's design and architecture:

### 1 - Extraction:

- The *extract.py* module includes functions for fetching data from an API and saving it as a JSON file.
  - The *fetch_data_from_api(api_url)* function makes a GET request to the specified API URL using the requests library and retrieves the JSON response.
  - The *extract(api_url)* function calls fetch_data_from_api() to fetch data from the API and returns it.
  - The *save_to_json(data)* function writes the result to a JSON file named *"fipiran_data.json"*.

### 2- Transform:

- The *transform.py* module focuses on transforming the extracted data into a structured format suitable for analysis.
  - The *load_json_file(name)* function loads data from the specified JSON file and returns it.
  - The *main()* function loads the data from "fipiran_data.json", creates a pandas DataFrame from the 'items' key in the loaded data, and performs necessary transformations on the DataFrame.
  - The transformations include filling NaN values, converting the 'initiationDate' column to datetime format, and interpolating missing values.
  - The resulting transformed DataFrame is returned.

### 3 – Load:

- The *load.py* module handles the loading of the transformed data into separate tables or CSV files.
  - The *drop_Nan_columns(df)* function drops any columns with all NaN values from a DataFrame. (Please note that this function must be placed in the

"Transform" section. However, for easier handling, the dropping operation was implemented in the "Load" section.)

- The *main(df)* function takes the transformed DataFrame as input and creates separate sub-dataframes for different tables in the database.
- Each sub-dataframe corresponds to a specific table, such as rank, fund, efficiency, manager, and detail.
  As mentioned in the section 4, the sub-dataframes saved as separate CSV files in the 'database' directory.

## Deployment

- At first, installing the required libraries is needed. Running the command:
  pip install -r requirements.txt
  in your terminal, will install the packages listed in the requirements.txt file, ensuring that you have the same package versions as specified.
- Execute the pipeline by running the ETL.py script, which orchestrates the extraction, transformation, and loading steps.

## Remarkable Points:

- Due to the lack of time, the section 5 and running a database for section 4 hadn't done. And as mentioned in last sections, some folders and sub-dataframes are used as a relational database.
- I tried to get good information about the data in the available time. However, there may be errors in the classification of data in the tables.