**Systems**:

* **Exercise**: Setup a Docker Stack
* **Expected results**:
  + Provide 2 docker files (not docker images)
  + Provide 1 docker-compose file that will be used to launch and bind all the images
  + Provide expected folder structure the docker images require
* **Description**:
  + Create a docker stack comprising of two docker images **dev**, and **db**. The images should be linked to each other and should be able to use services deployed. As an example, the dev image should be able to connect to SQL database which is present in the DB image. Images must be built on **Ubuntu 20.04**. Details of the two images are here:
  + Dev Image:
    - R studio server
    - Jupyter lab
    - User when logged into the image, should have sudo access
  + DB Image:
    - MS SQL Server 2019 (Linux flavour)
      * Setup appropriate logins
      * Make sure the data saved in the SQL server is not lost when the image restarts

**ETL**:

* **Exercise**: Write an ETL script in R
* **Expected Results**: Working R script, data saved and persisted in sql server
* **Details**:
  + Expected Write R script to perform an ETL job, the job should run without any parameters. It needs to be executed inside the R studio image created above. The aim of the script is to check for any new data available, if found, download the data, and save it inside the SQL server running in the **DB Image** docker container.
  + Website:
    - <https://www.invesco.com/us/financial-products/etfs/product-detail?audienceType=Investor&ticker=BKLN>
  + The link above has many tables, the following need to be extracted on daily basis. An example image of the table is shown in the screen shot below
    - BKLN Intraday Stats
    - Yield
    - Prior Close
    - Fund Characteristics
  + Graphical user interface, text, application, email

    Description automatically generated

**Snow-SQL**:

* **Exercise**: Understand the two tables shown in the images below. Using the understanding of the schema and the columns present in the images, write an Snow-SQL query. The query will join two table and compute an aggregate. There are two tables viz. prices and dividends. The prices table is shown in first image and has p\_price column. The dvidiends table is shown in second image and has p\_divs\_pd column. We need to compute the total of p\_divs\_paid for each row in price table where the price.p\_date < dividends.p\_divs\_exdate. In summary this means compute total dividend paid after each price date. The output of the query will produce following columns: price.fsym\_id, price.p\_date, sum( dividend.p\_divs\_pd )
* **Expected Result**: SQL query
* **Description**:
  + The query works on two tables namely Dividends and Prices
  + Prices has following data:
  + Table

    Description automatically generated
  + And Dividends has following data:
  + Table

    Description automatically generated
* We need to calculate total dividends paid after each price date
* This translates to sum of [Dividends.p\_divs\_pd] for each Prices.FSYM\_ID and Dividvidends.p\_divs\_exdate > Price.p\_date

Git:

* **Exercise**: Operate github account
* **Expected results**: We expect access to the all the files that were generated in all previous parts of this exercise checked into a public github repository and shared to us. We will clone the repository to access all the results.
* **Description**: All code generated in above parts of this exercise should be pushed into a new repository on any github account, and access to the repository provided to us. The files we expect are:
  + Docker/Docker-compose files
  + R script
  + SQL script