**Assignment 4**

Yuval Bubnovsky – 311252415, Talia Seada – 211551601

[GitHub Repository](https://github.com/TaliaSeada/DataBases_Ex4)

Question 1:

1. We did this question in SQL and in Spark functions:

Spark:

>>> df = sqlContext.read.option("multiline", "true").json("books.json")

>>> start\_F = start\_F.select("title", "author", "year")

>>> start\_F = start\_F.withColumn("number of years since published", 2022 - start\_F.year)

>>> start\_F = start\_F.filter("author like 'F%'")

>>> final = start\_F.select("title", "author", "number of years since published")

>>> final.show()Graphical user interface, text

Description automatically generated

Spark Explanation:

First line of code allows us to treat the JSON data as a table and apply functions and queries to it in a convenient way, we then select the three columns which are relevant to the question.

Next, we calculate how many years have passed since the book was written using withColumn(), filter the results to authors whose name starts with “F”, re-select the required columns and show the results.

SQL:

>>> df.registerTempTable("f\_table")

>>> start\_with\_f = sqlContext.sql("SELECT title, author, (2022 - year) as `number of years since published` FROM f\_table WHERE author LIKE 'F%'")

>>> start\_with\_f.show()

Text

Description automatically generated

SQL Explanation:

Same we as saw in the lecture.

>>> df = sqlContext.read.option("multiline", "true").json("books.json")

>>> english\_books = df.filter("language LIKE 'English'")

>>> english\_books = english\_books.groupBy("author").avg("pages")

>>> english\_books.show()

A picture containing text

Description automatically generated

Explanation:

First line of code allows us to treat the JSON data as a table and apply functions and queries to it in a convenient way, then we take only the English books, we group by the 'author' to know the information about each author, then we calculate the average of the number of pages each author wrote.

Question 2:

1. Main.py:  
   First, we dealt with the data since we got a text file (and not a csv file) we had to set it. For each line we split by a comma, then we make each value of type float (we did it twice, for the linear and for the logistic).

Text

Description automatically generated

Then we ran the linear regression model we built.  
For the linear regression we split the data to 75% train set and 25% test set, then we trained the model on the training set and at the end we sent the test set to the trained model and got the MSE (loss).

Text

Description automatically generated

Results:  
Blue – predicted, Orange – actual:

Chart, line chart

Description automatically generated

Text

Description automatically generated

Text

Description automatically generatedAfter that, we ran the logistic regression model we built.  
For the logistic regression we split the data to 85% train set and 15% test set, then we trained the model on the training set and at the end we sent the test set to the trained model and got the F Measure results.

Results:

Blue – predicted, Orange – actual:

Text

Description automatically generatedChart, line chart

Description automatically generated

1. linear\_regression.py

In this class we built the linear regression model with the following functions:

* gradients – calculates the gradient.
* loss – calculates the MSE.
* fit – main function, trains on the train set.
* predict – calculates the MSE after running the trained mode on the test set.

1. logistic\_regression.py

In this class we built the logistic regression model with the following functions:

* gradients – calculates the gradient.
* loss – calculates the MSE.
* fit – main function, trains on the train set.
* accuracy – calculates the accuracy by the following function:

Text

Description automatically generated

* recall – calculates the recall by the following function:

A picture containing diagram

Description automatically generated

* precision – calculates the precision by the following function:

A picture containing text

Description automatically generated

* f\_score - calculates the F score by the following function:

Text

Description automatically generated

* predict – calculates the accuracy, recall, precision, and the F Measure.