**Vidzeme University of Applied Sciences**

**Faculty of Engineering**

Introduction to Python and Data Exploration

**group 06**

**practical work #6**

Valmiera, 2024

**Table of Contents**

[1.0 Content 4](#_Toc167116751)

[2.0 Tasks and their solutions 5](#_Toc167116752)

[2.1 The Task 5](#_Toc167116753)

[3.0 References 7](#_Toc167116754)

[4.0 Appendix (Code) 8](#_Toc167116755)

[4.1 Task\_6.1\_Gr.06.py 8](#_Toc167116756)

|  |  |  |  |
| --- | --- | --- | --- |
| Document versions | | | |
| Version | Status / Changes | Date | Author |
| 0.1 | First version implementation | 20.05.2024 | Kristers Kalniņš |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Contacts and responsible (-s) | | | |
| Name Surname | Department | Position | Contact information (e-mail) |
| Emīls Konrāds | Group #06 | Coordinator | emils.konrads@va.lv |
| Gatis Jurisons | Group #06 | Member | gatis.jurisons@va.lv |
| Kristers Kalniņš | Group #06 | Member | kristers.kalnins@va.lv |
|  |  |  |  |
|  |  |  |  |

# Content

**Goal:** The team (further referenced as team 06) must complete the given task by the client (university professor) described in tasks 1-3 of unit 4.

**Content**: 620 words, two images, two appendixes.

# Tasks and their solutions

Team 06 had organized a meeting after being given the task. The team had started organizing a workflow and work management chain during the weekend. The team 06 decided to split the workload in 2 different categories:

* Documentation
* Coding

## The Task

The task was to write a Python program that:

* Creates an SQLite database with at least 3 tables for storing email addresses, domain names, weekdays, and SPAM confidence levels, with foreign key relationships to avoid duplication.
* Reads data from mbox-short.txt to populate these tables.
* Prints a list of unique domains from the database.
* Allows the user to input a domain and displays emails received from that domain, formatted with the weekday, domain name, email address, and SPAM confidence level.
* Queries and displays emails received specifically on Fridays and Saturdays, formatted in four columns: weekday, domain name, email address, and SPAM confidence level.

A screenshot of a computer screen

Description automatically generated

Image: 1.1. first part of the code (Task\_6.1\_Gr.06.py)

A screen shot of a computer

Description automatically generated

Image: 1.2. second part of the code (Task\_6.2\_Gr.06.py)

# References

Osis, K. (2024, Mar). *Moodle.va.lv.* Retrieved from

https://moodle.va.lv/pluginfile.php/32208/mod\_resource/content/5/2024\_practical\_work\_6.pdf

Osis, K. (2024, Mar). *Moodle.va.lv.* Retrieved from Template of Document: https://moodle.va.lv/mod/resource/view.php?id=11925

# Appendix (Code)

## Task\_6.1\_Gr.06.py

import sqlite3

import re

# Function to extract domain from an email address

def extract\_domain(email):

    return email.split('@')[1]

# Connect to SQLite database (or create it if it doesn't exist)

conn = sqlite3.connect('email\_spam.db')

cursor = conn.cursor()

# Create tables if they don't exist

cursor.execute('''

CREATE TABLE IF NOT EXISTS email\_address (

    id INTEGER PRIMARY KEY AUTOINCREMENT,

    email TEXT NOT NULL UNIQUE,

    weekday TEXT

)

''')

cursor.execute('''

CREATE TABLE IF NOT EXISTS domain\_name (

    id INTEGER PRIMARY KEY AUTOINCREMENT,

    domain TEXT NOT NULL UNIQUE

)

''')

cursor.execute('''

CREATE TABLE IF NOT EXISTS spam\_confidence (

    id INTEGER PRIMARY KEY AUTOINCREMENT,

    email\_id INTEGER NOT NULL,

    confidence\_level REAL,

    FOREIGN KEY (email\_id) REFERENCES email\_address(id)

)

''')

conn.commit()

# Read the mbox file

with open('mbox-short.txt', 'r') as file:

    lines = file.readlines()

# Regular expressions for matching email addresses and spam confidence levels

email\_pattern = re.compile(r'^From (\S+@\S+)')

spam\_confidence\_pattern = re.compile(r'^X-DSPAM-Confidence: ([0-9.]+)')

email = None

weekday = None

spam\_confidence = None

for line in lines:

    # Match email addresses

    email\_match = email\_pattern.match(line)

    if email\_match:

        email = email\_match.group(1)

        # Extract weekday if available

        parts = line.split()

        if len(parts) >= 3:

            weekday = parts[2]

        # Insert email and weekday into the email\_address table if not exists

        cursor.execute('INSERT OR IGNORE INTO email\_address (email, weekday) VALUES (?, ?)', (email, weekday))

        cursor.execute('SELECT id FROM email\_address WHERE email = ?', (email,))

        email\_id = cursor.fetchone()[0]

        # Extract domain and insert into the domain\_name table if not exists

        domain = extract\_domain(email)

        cursor.execute('INSERT OR IGNORE INTO domain\_name (domain) VALUES (?)', (domain,))

    # Match spam confidence levels

    spam\_confidence\_match = spam\_confidence\_pattern.match(line)

    if spam\_confidence\_match and email:

        spam\_confidence = float(spam\_confidence\_match.group(1))

        # Insert spam confidence into the spam\_confidence table

        cursor.execute('INSERT INTO spam\_confidence (email\_id, confidence\_level) VALUES (?, ?)', (email\_id, spam\_confidence))

        email = None  # Reset email to avoid duplicate entries in the next iteration

conn.commit()

# Fetch and display unique domains

cursor.execute('SELECT DISTINCT domain FROM domain\_name')

unique\_domains = cursor.fetchall()

print("List of unique domains:")

for domain in unique\_domains:

    print(domain[0])

# Get user input for domain selection

chosen\_domain = input("\nEnter a domain name from the list above: ").strip()

# Query the database for emails from the chosen domain on Fri and Sat, ordered by spam confidence level

query = '''

SELECT DISTINCT email\_address.weekday, domain\_name.domain, email\_address.email, spam\_confidence.confidence\_level

FROM email\_address

JOIN domain\_name ON email\_address.email LIKE '%' || domain\_name.domain

JOIN spam\_confidence ON email\_address.id = spam\_confidence.email\_id

WHERE domain\_name.domain = ?

AND email\_address.weekday IN ('Fri', 'Sat')

ORDER BY spam\_confidence.confidence\_level

'''

cursor.execute(query, (chosen\_domain,))

results = cursor.fetchall()

# Display the results

print("\nResults for emails from domain '{}':".format(chosen\_domain))

for row in results:

    print("Weekday: {}, Domain: {}, Email: {}, SPAM Confidence: {}".format(row[0], row[1], row[2], row[3]))

# Close the database connection

conn.close()