Laurel Technology Solutions Project

A special thank you for your invaluable guidance and support.

- **Mr Ashley Williamson**

YOUSSEF IBOURK

INTRODUCTION :

In this semester, I had explored and gained knowledge about many things relating to data science and python. Now, I am taking the next step and delving deeper into the subject matter. I am expanding upon my knowledge and work by developing a Python code to propose solutions for the data integration and management project for Laurel Technology Solutions Ltd, a small-to-medium business. The company, as a part of its expansion plan, is looking to further investigate and exploit their data, but currently, the customer data is stored in various IT systems, in different formats such as xml, json, csv, and txt. This makes it difficult for the company to have a single cohesive record representing all of their customer data. The main objective of this project is to combine these data files into one csv file and then upload this file to a database.

The proposed solution involves developing an ETL (Extract, Transform, Load) process to extract the data from the different systems, transform the data into a standardized format, and load it into the central database. This will allow for a unified view of customer data and support the company's goals of data analysis and exploitation. The report will provide a detailed description of the proposed solution, including the specific technologies and tools that will be used, a detailed project plan, and an implementation schedule. It will also discuss the potential challenges and risks that may be encountered during the implementation of this project and the steps that will be taken to mitigate them. I will also include a critical evaluation of the proposed solution, including a cost-benefit analysis and a discussion of how the solution aligns with the company's goals and objectives.

STEPS :

In order to fulfil the company's objecties, we need to:

- First of all, extract the data from the different systems and formats: Using the standard libraries, which is the biggest challenge of the project, such as xml.etree.ElementTree, json, and csv, you can extract the data from the different systems and formats. The xml.etree.ElementTree library can be used to extract the data from xml files, json library can be used to extract data from json files, and csv library can be used to extract data from csv and for txt files, the first thing that comes to mind is to use the "re" library to extract useful informations such as, names and prices.
- Then, transform the data into a standardized format, which means, once we have extracted the data, it's important to standardize the data format. This can be done by converting all the

data into a list of dictionaries, where each dictionary represents a record in the data. This makes it easier to work with the data and ensures consistency across the different formats.

- The next step is to perform data cleaning and preprocessing, this can include removing missing values, handling duplicate records, and transforming data into a format that is suitable for the database. This step is important to ensure the data is accurate and consistent, and to avoid any errors when loading the data into the database.
- And after that comes the step of writing the cleaned and preprocessed data to a csv file. Once the data is cleaned and preprocessed, it can be written to a csv file using the csv library. This step will be used to upload the data to the database later.
- And here comes one of the most essentials steps, which is to connect to the database and create the necessary tables and schema: Using Pony ORM library, we can connect to the database and create the necessary tables and define the schema for the data. This step is important to ensure the data is structured and organized in the database.
- And finally, we will have to load the data into the database: Using the Pony ORM library to load the data from the csv file into the database, we can use the `import_from_csv` method of thePony ORM library to easily and efficiently load the data from the csv file into the database. This method allows us to specify the location of the csv file and the table that we want to import the data into. We can also set options such as the delimiter used in the csv file and the column names to use when creating the table.

PROBLEMS AND SOLUTIONS :

The fragmentation of customer data across various IT systems and in different formats such as xml, json, csv, and txt can cause several problems for the company. One of the main issues is that it makes it difficult for the company to have a single cohesive record representing all of their customer data, leading to inefficiencies in data management and analysis. For example, if the company wants to run a query on customer data, it would have to manually extract data from multiple systems, which can be time-consuming and prone to errors.

Furthermore, the lack of a centralized view of customer data can make it challenging for the company to gain valuable business insights and make data-driven decisions. It can also lead to a lack of consistency and accuracy in the data, which can impact the quality of the analysis and the ability to make accurate predictions. Additionally, it can also hinder the company's ability to improve customer engagement, as it would be difficult to segment and target customers based on their data.

And here comes the common problems that they could face as a company:

- Data duplication and inconsistencies:

They can have a significant impact on the accuracy and reliability of the data. When the same data is stored in multiple systems, it can become difficult to maintain consistency and accuracy.

One of the main problems with data duplication is that it can lead to errors and inconsistencies in the data. For example, if a customer's information is stored in multiple systems, and one system is updated while the other is not, this can lead to inconsistencies between the two systems. This can cause confusion and lead to incorrect decisions being made based on the data.

To avoid these issues, it's important to establish processes and systems to ensure that data is entered and updated consistently across all systems, and that duplicate data is identified and removed. Additionally, regular data quality checks and audits can help to identify and resolve any data inconsistencies.

- Difficulty in data analysis and reporting:

can have a major impact on a company's ability to make data-driven decisions and track key performance indicators. When data is stored in different formats and systems, it can be challenging to extract, clean, and analyze the data effectively.

One of the main problems with data stored in different formats is that it can make it difficult to integrate and combine the data. For example, if customer data is stored in different systems in different formats such as CSV, JSON, and XML, it can be challenging to combine and analyze the data in a meaningful way. This can lead to a lack of actionable insights and make it difficult to track key performance indicators.

To overcome these issues, it is important to establish processes and systems that allow for data to be extracted, cleaned, and integrated from different systems and formats. This can be done using standard libraries or specialized data integration tools. Additionally, regular data quality checks and audits can help to identify and resolve any data inconsistencies. By having a centralized data store, it becomes easier to perform data analysis and reporting, which can lead to a more accurate understanding of key performance indicators and more actionable insights.

- Difficulty in customer data management:

Difficulty in customer data management can have a major impact on a company's ability to effectively target and segment customers. When customer data is stored in different systems, it can be challenging to manage and maintain accurate customer information.

One of the main problems with having data stored in different systems is that it can make it difficult to maintain a complete and accurate customer profile. For example, if customer data is stored in different systems such as financial systems, HR systems, and marketing systems, it can be challenging to keep all of the customer information up to date and accurate. This can lead to a lack of accurate customer information and make it difficult to target and segment customers effectively.

To overcome these issues, it is important to establish processes and systems that allow for customer data to be consolidated and maintained in a centralized location. This can be done using standard libraries or specialized data integration tools. Additionally, regular data quality checks and audits can help to identify and resolve any data inconsistencies. By having a centralized data store, it becomes easier to manage customer data effectively, which can lead to a more accurate understanding of customer behavior and preferences and more targeted marketing campaigns.

- Difficulty in data sharing and collaboration:

Having data stored in different systems can make it difficult for employees to share and collaborate on data. This can lead to delays in decision making and hinder the effectiveness of the team.

- Difficulty in data migration:

If the company decides to change its IT systems, it can be difficult and time consuming to migrate the data from the old systems to the new ones due to the different formats and systems the data is currently stored in. This can lead to data loss or corruption during the migration process, which can negatively impact the company's operations.

One possible solution to overcome the difficulty in data migration is to establish a data migration plan. This plan should include some steps, like a **Data inventory** to identify all of the systems and formats that the data is currently stored in so that the company can ensure that the data migration process is carried out smoothly and that data loss and corruption is minimized. It's also important to test the migration process in a testing environment before applying it to the production systems.

In summary, the fragmentation of customer data can lead to inefficiencies in data management, which make it difficult to gain valuable business insights and make data-driven decisions, and hinder the company's ability to improve customer engagement.

ADVANCED SOLUTIONS :

Some recommendations that Laurel Technology Solutions Ltd. could consider include:

- Data warehousing and business intelligence (BI) tools: These technologies will allow the company to centralize their customer data and create a single cohesive record. They can then use BI tools to analyze and exploit the data for business insights and decision making.
- Cloud-based solutions: As the company plans to expand to foreign markets, a cloud-based solution will allow for easy access to customer data from different locations. Additionally, it will enable seamless communication and collaboration between the UK and South Korean offices.
- Data integration and ETL (Extract, Transform, Load) tools: These technologies will enable the company to combine their different data streams and integrate them into a central database. This will allow for a unified view of customer data and will support the company's data analysis and exploitation goals.
- Cybersecurity solutions: As the company will be handling sensitive customer data, it is important to ensure that the data is protected from cyber threats. The company should invest in cybersecurity solutions such as firewalls, intrusion detection and prevention systems, and data encryption.
- Virtual Private Network (VPN): To ensure secure communication between the UK and South Korean offices, a VPN can be used to encrypt the data being transmitted over the internet.
- Customer Relationship Management (CRM) software: To manage their expanding customer base, the company could consider investing in a CRM system. This will allow the company to track customer interactions, manage sales and marketing activities, and improve customer engagement.
- Social media management tools: As the company plans to expand their offerings and operations to include more social media aspects, it may be beneficial to invest in social media management tools. This will allow the company to monitor social media activity and engage with customers on social media platforms.
- Project management software: As the company expands and takes on more customers, it may be beneficial to invest in project management software. This will allow the company to manage projects, collaborate with team members, and track progress and deadlines.
- Data visualization tools: These tools will enable the company to present their data in an easy-to-understand format, allowing for better decision making and data-driven insights.

- Remote working tools: As the company expands to foreign markets, it may be beneficial to invest in remote working tools to enable employees in different locations to collaborate and communicate effectively. These tools may include video conferencing, instant messaging, and file sharing software.

A critical evaluation of these technologies would be to consider the costs, scalability, ease of use, and data security. The company should also consider the potential benefits of these technologies and how they align with their business goals. Additionally, it is important to consider the available resources and skillsets within the company to ensure that the implementation of these technologies is successful.

FUTURE DEVELOPMENTS :

Once the data is loaded into the database, we can use Pony ORM's querying capabilities to perform various operations on the data, such as retrieving, updating and deleting records, and that is one of the main purposes of this project.

It is important to keep in mind that, depending on the amount of data we have, loading the data into the database may take some time, so we should consider implementing some kind of progress indicator or status message to let the user know the progress of the process and to avoid the user thinking that the program is not working.

- Data validation: To add data validation, we can create a set of validation rules and check them against the data before inserting it into the database. This can include checks such as ensuring that required fields are not empty, data is in the correct format, and that data falls within a specified range. For example, we can use regular expressions to check that an email address is in the correct format, or we can check that a date falls within a certain range.

- Data security: To add data security, we can implement encryption and authentication techniques. Encryption can be used to protect sensitive data such as credit card numbers or personal identification numbers, (luckily we are working with fake informations). Authentication can be used to ensure that only authorized users have access to the data. We can also add a role-based access control system, which allows us to restrict access to the data based on user roles.

- Data visualization: To add data visualization capabilities, we can use libraries such as Plotly, Chart.js, or Datawrapper to create interactive charts, plots, and maps. These libraries offer a

wide range of chart types and customization options, allowing us to create visually appealing and informative visualizations of the data.

- Data backup: To add data backup capabilities, we can schedule regular backups of the data to an external storage device or cloud-based storage service. This can be done using a cron job or other scheduling tool. we can also set up a backup system that automatically backs up the data when certain conditions are met, such as when the data is updated or when the database reaches a certain size.

- Error handling: To add error handling capabilities, we can use try-except statements to catch unexpected errors and exceptions. This will help to ensure that the program does not crash and can continue to operate normally even in case of an error. We can also use logging to record any errors that occur, which can be useful for debugging and troubleshooting.

- Automation: To add automation capabilities, we can schedule the data processing, loading, and backups on regular intervals. This can be done using a cron job or other scheduling tool. This can save a lot of time and effort for the users as they don't have to manually perform these tasks.

- Scalability: To add scalability features, we can optimize the database design for performance, use distributed architecture, and use cloud-based services to handle larger data sets and a growing number of users. We can also use caching and indexing to improve performance.

- Reporting and Analytics: To add reporting and analytics capabilities, we can use libraries such as pandas, numpy, and matplotlib…

CONCLUSION :

In conclusion, this project aimed to combine data files from different formats (xml, json, csv, and txt) into a single csv file and then upload this file to a database such as phpmyadmin. The goal was to have a central location for the data in order to unify and pool the data together for further data investigation and exploitation. The project's steps included: reading the data from the different formats using standard libraries, combining the data into one csv file, and then loading the data into the database using ponyorm library.

The project addressed several challenges that the company was facing, such as difficulty in data analysis and reporting, customer data management, and data migration. Combining the data into one csv file and uploading it to a database helped to overcome these challenges by allowing the company to have a single cohesive record representing all of their customer data, which can be

easily analyzed, and reported on. Additionally, the project helped to improve the company's core infrastructure by combining their different data streams with an aim to analyze and exploit their data.

As for later versions, the project may include more advanced data analysis and visualization tools, data quality checks, and data archiving. This will help to improve the accuracy and reliability of the data and allow the company to make better data-driven decisions. Overall, the project has been a success in consolidating the company's customer data, which will help the company to make better use of their data, and support their expansion into new markets.

REFERENCES :

- Bansal, S.K. and Kagemann, S., 2015. Integrating big data: A semantic extract-transform-load framework. *Computer*, *48*(3), pp.42-50.
- Cielen, D. and Meysman, A., 2016. *Introducing data science: big data, machine learning, and more, using Python tools*. Simon and Schuster.
- DataCamp. (2021). DataCamp website. [online] Available at: https://www.datacamp.com/ (Accessed: January 20, 2023)
- Dataquest. (2021). Dataquest website. [online] Available at: https://www.dataquest.io/ (Accessed: January 20, 2023)
- Ibarrera. (2021, November 15). Merging data from multiple sources. [Online]. Available at: https://dataladder.com/merging-data-multiple-sources/ (Accessed: date accessed)
- Ilyas, I.F. and Chu, X., 2015. Trends in cleaning relational data: Consistency and deduplication. *Foundations and Trends® in Databases*, *5*(4), pp.281-393.
- Kache, F. and Seuring, S., 2017. Challenges and opportunities of digital information at the intersection of Big Data Analytics and supply chain management. *International journal of operations & production management*.
- Khuller, S., Kim, Y.A. and Wan, Y.C., 2003, June. Algorithms for data migration with cloning. In *Proceedings of the twenty-second ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems* (pp. 27-36).
- Liang, T.P. and Liu, Y.H., 2018. Research landscape of business intelligence and big data analytics: A bibliometrics study. *Expert Systems with Applications*, *111*, pp.2-10.
- Müller, A.C. and Guido, S., 2016. *Introduction to machine learning with Python: a guide for data scientists*. " O'Reilly Media, Inc.".
- Pony ORM. (2022). Pony ORM documentation. [online] Available at: https://docs.ponyorm.org/ (Accessed: January 20, 2023).
- Regola, N. and Chawla, N.V., 2013. Storing and using health data in a virtual private cloud. *Journal of medical Internet research*, *15*(3), p.e2076.

- Wang, A.Y., Mittal, A., Brooks, C. and Oney, S., 2019. How data scientists use computational notebooks for real-time collaboration. *Proceedings of the ACM on Human-Computer Interaction*, *3*(CSCW), pp.1-30.
- Yulianto, A.A., 2019. Extract transform load (ETL) process in distributed database academic data warehouse. *APTIKOM Journal on Computer Science and Information Technologies*, *4*(2), pp.61-68.

R-CODE :

1 :

```
-----------------

# JSON TO CSV
import json
import csv
# open the json file in read mode
with open(r'D:\modules\Msc_Data_Science\Data_Science_Fundamentals\CETM50\assignment\
data = json.load(json_file)

# create a csv file
json_file_write = open(r'D:\modules\Msc_Data_Science\Data_Science_Fundamentals\CETM5
csv_writer = csv.writer(json_file_write)

# counter to iterate over the csv file and write the first line (header)
counter = 0
for idx in data:
    if counter == 0:
        header = idx.keys()
        csv_writer.writerow(header)
        break

# for loop to fill the csv file with values from the json file
for idx in data:
    if counter == 0:
        counter += 1
    csv_writer.writerow(idx.values())

# close the csv file
json_file_write.close()
```

```
# open the CSV file in read mode
with open(r'D:\modules\Msc_Data_Science\Data_Science_Fundamentals\CETM50\assignment\

# create a CSV reader object
reader = csv.reader(json_file_read)

# read the contents of the file into memory
labels = list(reader)

# loop to iterate over the json file to check for values that are = "amount"
for line_no , line in enumerate(json_file_read):
    # line hwwa lcontenu
    # w line_no hwwa indice dyal ligne
    if "amount" in line.lower():
        line_num.append(line_no)

# create two lists to store their values
value_amount = []
value_time_period_years = []

# for loop to store values that are "values" in the first list and those that are "t
for idx in line_num:
    dictt = data[idx-1]

    value_amount.append(dictt["debt"]["amount"])
    value_time_period_years.append(dictt["debt"]["time_period_years"])

# delete the old values (first version of jsontocsv file)
for i, row in enumerate(labels):
    for j, cell in enumerate(row):
        if 'amount' in cell:
            # Delete the value from the cell
            labels[i][j] = ''

# Add the new column name to the header row
labels[0].append('debt')
labels[0].append('debt_amount')
labels[0].append('debt_time_period_years')

#Open the CSV file in write mode
with open(r'D:\modules\Msc_Data_Science\Data_Science_Fundamentals\CETM50\assignment\

# Create a CSV writer object
writer = csv.writer(data_file_write)
```

```
# Write the rows back to the file
for row in labels:
    writer.writerow(row)


for line_no , line in enumerate(data[line_no][12]):
    if line_no == line_num:
        writer.writerow(value_amount[line_no])
        writer.writerow(value_time_period_years[line_no])
```

2:

```
# SORT JSON_TO_CSV FILES TO MERGE THEM
import csv

# open the CSV file in read mode
with open(r'D:\modules\Msc_Data_Science\Data_Science_Fundamentals\CETM50\assignment\

# create a CSV reader object
reader = csv.reader(csv_file_read)

# read the rows of the CSV file into a list, excluding the first row
rows = list(reader)[1:]

# Get the index of the column that i want to sort by
with open(r'D:\modules\Msc_Data_Science\Data_Science_Fundamentals\CETM50\assignment\
    col_names = next(csv.reader(csv_file_read))

# to choose which column name i want to sort my file in alphabetical order
sort_by_last_name = col_names.index('lastName')

# take an argument row, and return the value at a specific index of that row, specif
rows.sort(key=lambda row: row[sort_by_last_name])

# insert the first row back into the list at the beginning
with open(r'D:\modules\Msc_Data_Science\Data_Science_Fundamentals\CETM50\assignment\
    line = next(csv.reader(csv_file_read))
rows.insert(0, line)

# create a new csv file where we will have the sorted values
with open(r'D:\modules\Msc_Data_Science\Data_Science_Fundamentals\CETM50\assignment\

writer = csv.writer(csv_file_write)
```

```
# Write the sorted rows back to the CSV file
writer.writerows(rows)
```

3:

```
# XML TO CSV
from xml.etree import ElementTree
import csv

# to navigate the content of the XML file.
xml = ElementTree.parse(r"D:\modules\Msc_Data_Science\Data_Science_Fundamentals\CETM

# create the csv file where we will put the conversion of the xml file
with open(r"D:\modules\Msc_Data_Science\Data_Science_Fundamentals\CETM50\assignment\
csv_file_write_writer = csv.writer(csv_file_write)

# ADD THE HEADER TO CSV FILE
csv_file_write_writer.writerow(["address_postcode","commute_distance","company","pen

# search all elements with "user"
for user in root.findall('user'):
    address_postcode = user.get('address_postcode')
    commute_distance = user.get('commute_distance')
    company = user.get('company')
    pension = user.get('pension')
    salary = user.get('salary')
    marital_status = user.get('marital_status')
    dependants = user.get('dependants')
    retired = user.get('retired')
    sex = user.get('sex')
    age = user.get('age')
    lastName = user.get('lastName')
    firstName  = user.get('firstName')

    csv_line = [address_postcode, commute_distance, company, pension, salary, marita
    # print(csv_line)
    csv_file_write_writer.writerow(csv_line)
```

4:

```
# SORT XML_TO_CSV FILES TO MERGE THEM
import csv
```

```
with open(r'D:\modules\Msc_Data_Science\Data_Science_Fundamentals\CETM50\assignment\

reader = csv.reader(csv_file)

rows = list(reader)[1:]

with open(r'D:\modules\Msc_Data_Science\Data_Science_Fundamentals\CETM50\assignment\
    col_names = next(csv.reader(csv_file))
sort_by_last_name = col_names.index('lastName')

rows.sort(key=lambda row: row[sort_by_last_name])

with open(r'D:\modules\Msc_Data_Science\Data_Science_Fundamentals\CETM50\assignment\
    line = next(csv.reader(csv_file))
rows.insert(0, line)

with open(r'D:\modules\Msc_Data_Science\Data_Science_Fundamentals\CETM50\assignment\

writer = csv.writer(csv_file)

writer.writerows(rows)
```

5:

```
# SORT CSV FILE TO COMBINE
import csv

with open(r'D:\modules\Msc_Data_Science\Data_Science_Fundamentals\CETM50\assignment\

reader = csv.reader(csv_file)

rows = list(reader)[1:]

with open(r'D:\modules\Msc_Data_Science\Data_Science_Fundamentals\CETM50\assignment\
    col_names = next(csv.reader(csv_file))
sort_by_last_name = col_names.index('Second Name')

rows.sort(key=lambda row: row[sort_by_last_name])

with open(r'D:\modules\Msc_Data_Science\Data_Science_Fundamentals\CETM50\assignment\
    line = next(csv.reader(csv_file))
rows.insert(0, line)

with open(r'D:\modules\Msc_Data_Science\Data_Science_Fundamentals\CETM50\assignment\
```

```
writer = csv.writer(csv_file)

writer.writerows(rows)
```

6:

```
# MERGE THE 3 CSV FILES (CSV XML W JSON) IN ONE CSV FILE
import csv

# the first CSV file in read mode
with open(r'D:\modules\Msc_Data_Science\Data_Science_Fundamentals\CETM50\assignment\

# Create a CSV reader object
reader1 = csv.reader(csv_file1)

# Read the contents of the file into memory
lines1 = list(reader1)

# the second CSV file in read mode
with open(r'D:\modules\Msc_Data_Science\Data_Science_Fundamentals\CETM50\assignment\

# Create a CSV reader object
reader2 = csv.reader(csv_file2)

# Read the contents of the file into memory
lines2 = list(reader2)

# the third CSV file in read mode
with open(r'D:\modules\Msc_Data_Science\Data_Science_Fundamentals\CETM50\assignment\

# Create a CSV reader object
reader3 = csv.reader(csv_file3)

# Read the contents of the file into memory
lines3 = list(reader3)

# concatenate the columns from the three files
rows = []
for i in range(len(lines1)):
rows.append(lines1[i] + lines2[i] + lines3[i])

# open the new CSV file in write mode
with open(r'D:\modules\Msc_Data_Science\Data_Science_Fundamentals\CETM50\assignment\

# Create a CSV writer object
```

```
writer = csv.writer(csv_file_write)

# write the rows to the new file
for row in rows:
    writer.writerow(row)
```

7:

```
# TO DELETE THE DUPLICATED COLUMNS FROM THE COMBINED CSV FILE
import csv

with open(r'D:\modules\Msc_Data_Science\Data_Science_Fundamentals\CETM50\assignment\

reader = csv.DictReader(csv_file_read)

rows = list(reader)

# one to delete manually
column_name = ['firstName','age','lastName','sex']
for row in rows:
    for elt in column_name:
        del row[elt]
with open(r'D:\modules\Msc_Data_Science\Data_Science_Fundamentals\CETM50\assignment\

# get the key (column name) of the first row
# to delete automatically
fieldnames = list(rows[0].keys())

# check if any fieldname is repeated
for fieldname in fieldnames:
    if fieldnames.count(fieldname) > 1:
        # delete the repeated one
        fieldnames.remove(fieldname)

writer = csv.DictWriter(csv_file_write, fieldnames=fieldnames)

# Write the header row
writer.writeheader()

# Write the rows back to the file
for row in rows:
    writer.writerow({field: row[field] for field in fieldnames})
```

8:

```
# TO UPLOAD THE DATA TO THE DATABASE
from pony.orm import *
import csv

# Connect to the database
db = Database()
db.bind(provider='mysql', host='europa.ashley.work', user='student_bi27ty', passwd='

# make the mapping
column_name_mapping = {
    "First Name": "First_Name",
    "Second Name": "Second_Name",
    "Age (Years)": "Age",
    "sex": "Sex",
    "Vehicle Make": "Vehicle_Make",
    "Vehicle Model": "Vehicle_Model",
    "Vehicle Year": "Vehicle_Year",
    "Vehicle Type": "Vehicle_Type",
    "address_postcode": "address_postcode",
    "commute_distance": "commute_distance",
    "company": "company",
    "pension": "pension",
    "salary": "salary",
    "marital_status": "marital_status",
    "dependants": "dependants",
    "retired": "retired",
    "iban": "iban",
    "credit_card_number": "credit_card_number",
    "credit_card_security_code": "credit_card_security_code",
    "credit_card_start_date": "credit_card_start_date",
    "credit_card_end_date": "credit_card_end_date",
    "address_main": "address_main",
    "address_city": "address_city",
    "debt": "debt",
    "debt_amount": "debt_amount",
    "debt_time_period_years": "debt_time_period_years",
}

class Data_Frame(db.Entity):
    First_Name = Required(str)
    Second_Name = Required(str)
    Age = Required(int)
```

```
    Sex = Required(str)
    Vehicle_Make = Required(str)
    Vehicle_Model = Required(str)
    Vehicle_Year = Required(str)
    Vehicle_Type = Required(str)
    address_postcode = Required(str)
    commute_distance = Required(str)
    company = Required(str)
    pension = Required(str)
    salary = Required(str)
    marital_status = Required(str)
    dependants = Required(str)
    retired = Required(str)
    iban = Required(str)
    credit_card_number = Required(str)
    credit_card_security_code = Required(str)
    credit_card_start_date = Required(str)
    credit_card_end_date = Required(str)
    address_main = Required(str)
    address_city = Required(str)
    debt = Optional(str)
    debt_amount = Optional(str)
    debt_time_period_years = Optional(str)

# create the table in the database
db.generate_mapping(create_tables=True)
with open(r'D:\modules\Msc_Data_Science\Data_Science_Fundamentals\CETM50\assignment\
csv_reader = csv.reader(csv_file)
col_names = next(reader)
with db_session:
    # looping over each row in the csv_reader object
    for row in csv_reader:
        # mapping the column names
        data = Data_Frame(**{column_name_mapping.get(col_names[i], col_names[i]): ro
```

9:

```
# EXTRACTION OF useful informations from txt file
import csv
import re

# Read CSV file
with open(r'D:\modules\Msc_Data_Science\Data_Science_Fundamentals\CETM50\assignment\
reader = csv.DictReader(file)
first_names = []
```

```
second_names = []
names = []
for row in reader:
    first_names.append(row["First Name"])
    second_names.append(row["Second Name"])

with open(r'D:\modules\Msc_Data_Science\Data_Science_Fundamentals\CETM50\assignment\
text_lines = file.readlines()

# extraction of names and salaries from text file
for i, line in enumerate(text_lines):
text_names = line.split()
text_names = [x for x in text_names if x[0].isupper()]
text_names = [name.replace('"',' ').replace('!',' ').replace(',',' ').replace('\\','

# Check if names from text file match any names in CSV file
for name in text_names:
    if (name in first_names) or (name in second_names):
        print("Line", i+1, ": Name",name,"found.")

# Check if salary is in the line
salary = re.search(r'£\d+', line)
if salary:
    print("Line", i+1, ": Salary",salary.group(0),"found.")

percentage = re.findall(r'\d+\.\d+%', line)
if percentage:
    percentage = [float(val.replace("%", "")) for val in percentage]
    print("Line", i+1, ": Percentage of",percentage,"% found.")

# security code
sec_code = re.search(r'\"(\d{3})\"', line)
if sec_code:
    print("Line", i+1, ": Security code:", sec_code.group(1))
```