**Exploratory Data Analysis - Customer Loans in Finance**

You currently work for a large financial institution, where managing loans is a critical component of business operations.

To ensure informed decisions are made about loan approvals and risk is efficiently managed, your task is to gain a comprehensive understanding of the loan portfolio data.

Your task is to perform exploratory data analysis on the loan portfolio, using various statistical and data visualisation techniques to uncover patterns, relationships, and anomalies in the loan data.

This information will enable the business to make more informed decisions about loan approvals, pricing, and risk management.

By conducting exploratory data analysis on the loan data, you aim to gain a deeper understanding of the risk and return associated with the business' loans.

Ultimately, your goal is to improve the performance and profitability of the loan portfolio.

**Milestone 1: Setting up the environment**

In this scenario, you'll use GitHub to track changes to your code and save them online in a GitHub repo. Hit the button on the right to automatically create a new GitHub repo. We'll tell you when to use it as you go through the project.

**Milestone 2: Extract the loans data from the cloud**

A screenshot of a computer

Description automatically generated

**Task 1: Initialise a class to extract the data**

In this task you will initialise the class and script that you will use to extract the data from the cloud. The definition of the classes and methods are meant as guidance. If you would like to follow a difference project/class structure feel free to do so. The class methods won't be defined in this step, they will be defined once required in subsequent tasks.

Step 1: Create a new Python script db\_utils.py which will contain your code to extract the data from the database.

Step 2: Within the script create a new class called RDSDatabaseConnector. This class will contain the methods which you will use to extract data from the RDS database.

**Task 2: Extract the data from the RDS database**

The loan payments data is stored in an AWS RDS database. You will need to create the methods which will enable you to extract the data from this database.

Step 1: Create a credentials.yaml file to store the database credentials.

Remember to add this file to your .gitignore file in your repository, as you don't want your credentials being pushed to GitHub for security reasons.

Add the following credentials to your credentials.yaml file which will allow you to connect to the remote database:

RDS\_HOST: eda-projects.cq2e8zno855e.eu-west-1.rds.amazonaws.com

RDS\_PASSWORD: EDAloananalyst

RDS\_USER: loansanalyst

RDS\_DATABASE: payments

RDS\_PORT: 5432

Step 2: If you haven't already installed Python PyYAML package you should do so before the next step. This can be installed by running pip install PyYAML in the terminal and imported using import yaml. This will allow you to load your credentials.yaml file as a dictionary.

Step 3: After installing the package create a function which loads the credentials.yaml file and returns the data dictionary contained within. This will be be passed to your RDSDatabaseConnector as an argument which the class will use to connect to the remote database.

Step 4: Write the \_\_init\_\_ method of your RDSDatabaseConnector class. It should take in as a parameter a dictionary of credentials which your function from the previous step will extract.

Step 5: Define a method in your RDSDatabaseConnector class which initialises a SQLAlchemy engine from the credentials provided to your class. This engine object together with the Pandas library will allow you to extract data from the database.

Step 6: Develop a method which extracts data from the RDS database and returns it as a Pandas DataFrame. The data is stored in a table called loan\_payments.

Step 7: Now create another function which saves the data to an appropriate file format to your local machine. This should speed up loading up the data when you're performing your EDA/analysis tasks. The function should save the data in .csv format, since we're dealing with tabular data.

**Task 3: Familiarise yourself with the data**

With the data being stored locally, create a function which will load the data from your local machine into a Pandas DataFrame.

In this step you might want to print the shape of the data to understand the size of the data you're working with. Printing out a sample of the data can help give a quick overview of its columns and values. The columns names might be quite alien to you if you've never used data from the financial domain before. Below is a data dictionary which describes all columns of the database to help you understand their meaning:

id: Unique id of the loan

member\_id: Id of the member to took out the loan

loan\_amount: Amount of loan the applicant received

funded\_amount: The total amount committed to the loan at that point in time

funded\_amount\_inv: The total amount committed by investors for that loan at that point in time

term: The number of monthly payments for the loan

int\_rate (APR): Annual (APR) interest rate of the loan

instalment: The monthly payment owned by the borrower. This is inclusive of the interest.

grade: Loan company (LC) assigned loan grade

sub\_grade: LC assigned loan sub grade

employment\_length: Employment length in years

home\_ownership: The home ownership status provided by the borrower

annual\_inc: The annual income of the borrower

verification\_status: Indicates whether the borrowers income was verified by the LC or the income source was verified

issue\_date: Issue date of the loan

loan\_status: Current status of the loan

payment\_plan: Indicates if a payment plan is in place for the loan. Indication borrower is struggling to pay.

purpose: A category provided by the borrower for the loan request

dti: A ratio calculated using the borrower's total monthly debt payments on the total debt obligations, excluding mortgage and the requested LC loan, divided by the borrower’s self-reported monthly income

delinq\_2yr: The number of 30+ days past-due payments in the borrower's credit file for the past 2 years

earliest\_credit\_line: The month the borrower's earliest reported credit line was opened

inq\_last\_6mths: The number of inquiries in past 6 months (excluding auto and mortgage inquiries)

mths\_since\_last\_record: The number of months' since the last public record

open\_accounts: The number of open credit lines in the borrower's credit file

total\_accounts: The total number of credit lines currently in the borrower's credit file

out\_prncp: Remaining outstanding principal for total amount funded

out\_prncp\_inv: Remaining outstanding principal for portion of total amount funded by investors

total\_payment`: Payments received to date for total amount funded

total\_rec\_int: Interest received to date

total\_rec\_late\_fee: Late fees received to date

recoveries: Post charge off gross recovery

collection\_recovery\_fee: Post charge off collection fee

last\_payment\_date: Date on which last month payment was received

last\_payment\_amount: Last total payment amount received

next\_payment\_date: Next scheduled payment date

last\_credit\_pull\_date: The most recent month LC pulled credit for this loan

collections\_12\_mths\_ex\_med: Number of collections in 12 months' excluding medical collections

mths\_since\_last\_major\_derog: Months' since most recent 90-day or worse rating

policy\_code: Publicly available policy\_code=1 new products not publicly available policy\_code=2

application\_type: Indicates whether the loan is an individual application or a joint application with two co-borrowers