Project 2 – ETL Technical Report

A Case Study of the Extract, Transform, Load Process:

Credit Card Approval Predictions

Data of Submission: Monday 19th of December 2022

Robert Franklin, Sandra Reyes, Zalak Shah

Table of Contents

Table of Figures

Executive Summary

Commercial banks receive a lot of applications for credit cards. Many of them get rejected for many reasons, like high loan balances, low income levels, or too many inquiries on an individual's credit report, for example. Manually analyzing these applications is mundane, error-prone, and time-consuming (and time is money nowadays !).

In today's time luckily , this task can be automated with the power of machine learning and pretty much every commercial bank does so nowadays. In this project, companies are able to build an automatic credit card approval predictor using machine learning techniques, just like the real banks do.

Throughout the project, we worked closely as a team to design and implement the ETL process, using data analytics techniques we have learnt to code and test these steps as efficiently as possible.

We encountered several challenges and issues during the project, including data quality issues and difficulties in determining the best data sources. However, by collaborating closely and utilizing our technical skills and knowledge, we were able to overcome these obstacles.

Overall, the data analytics ETL project was a success and we believe that the database created will provide an excellent source for future analysis. We look forward to continuing to work on similar projects in the future and using our data analysis skills to provide valuable insights.

1. Introduction

Welcome to the technical report for our data analytics ETL project. In this report, we will provide a detailed overview of the steps taken to extract, transform, and load data from data sources into a database

The goal of this project was to obtain at least two data files from a reliable source, ideally in .csv or .json file type. The data source we access was credit card application data used to predict the credit application approvals (宋骁 Seanny 2019). We then used a combination of Postgres, Python, and other ETL frameworks to extract, transform, and load the data, while ensuring the accuracy and integrity of the resulting dataset. Throughout the project, we worked closely as a team to design and implement the ETL process. Using Jupyter Notebooks to ensure code was executed correctly, and fix errors were appropriate.

In this report, we will describe the various data sources used in the project, the tools and technologies employed, and the steps taken to extract, transform, and load the data. We will also provide examples of the challenges or issues encountered during the project.

2. Extract

The first step of the project was to analyse the data and create an entity relationship diagram (ERD) so to better visualise the data. This involved identifying the different column names, assigning the appropriate data types, and then identifying any primary or foreign keys. The resulting ERD is shown in Figure 1.

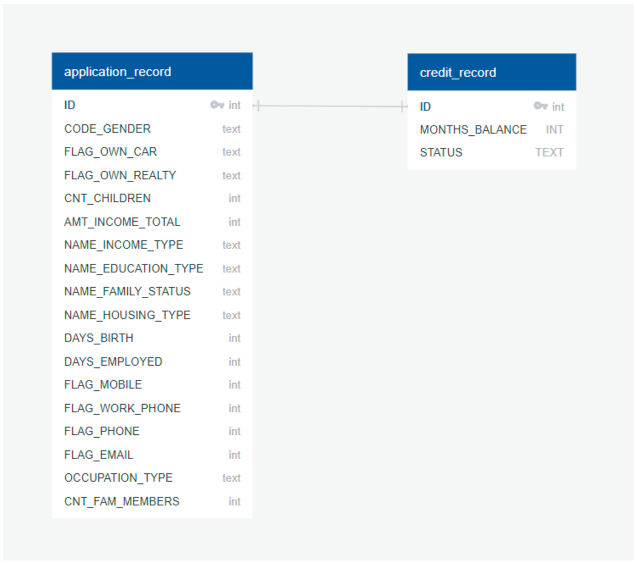
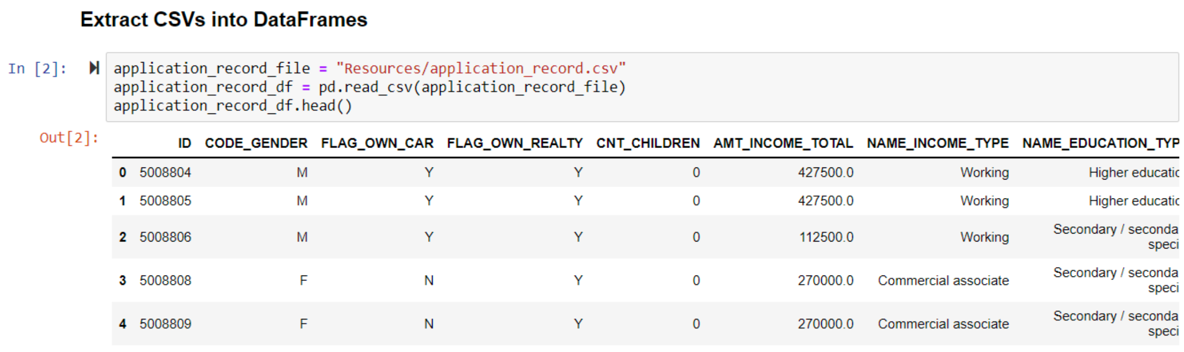


Figure 1 - Entity Relationship Diagram (ERD)

Following this, we set up the database in Postgres by first creating a new database called 'creditcard_db'. We decided to use Postgres as we wanted to create a relational database using SQL.

We then executed the SQL to create the schema to match the ERD. During this process we had some challenges with the primary key. We had incorrectly defined the 'ID' column as a primary key in the second csv file when it has the same ID listed several times to reflect multiple applications by the same applicant. Therefore, we changed this column to a foreign key.

Once the database and schema were created, we then created a new Jupyter notebook to extract the data into a Pandas DataFrame. Figure 2 below shows the first 5 rows of the first dataframe we named 'application_record_df'. It was important in this step to have our data source CSV files in a folder called Resources for ease of extraction.



Extract CSVs into DataFrames

```
In [2]:  application_record_file = "Resources/application_record.csv"
         application_record_df = pd.read_csv(application_record_file)
         application_record_df.head()
```

Out[2]:

	ID	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALTY	CNT_CHILDREN	AMT_INCOME_TOTAL	NAME_INCOME_TYPE	NAME_EDUCATION_TYP
0	5008804	M	Y	Y	0	427500.0	Working	Higher educatic
1	5008805	M	Y	Y	0	427500.0	Working	Higher educatic
2	5008806	M	Y	Y	0	112500.0	Working	Secondary / seconda speci
3	5008808	F	N	Y	0	270000.0	Commercial associate	Secondary / seconda speci
4	5008809	F	N	Y	0	270000.0	Commercial associate	Secondary / seconda speci

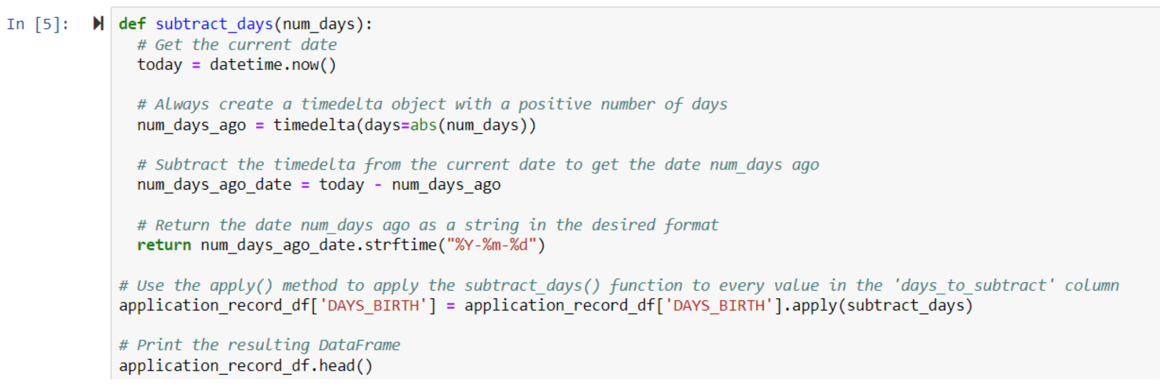
Figure 2 - Extracting CSV file into a Pandas DataFrame

3. Transform

Headers: During the process we found out that the headers were not easy to read , so we decided to change it, we did the research and finally we found out that was not a good idea to change as possible incur in an error later.

We transform the days of the "DAYS_BIRTH" and "DAYS_EMPLOYED" columns, to a "date format " to make it easy to read. We did this by importing the "datetime" and "timedelta" libraries, and by creating a formula called "subtract days" which we used to convert the existing format to datetime.

Figure 3 below shows the subtract_days formula coding, and its application using the apply() method to the "DAYS_BIRTH" column.



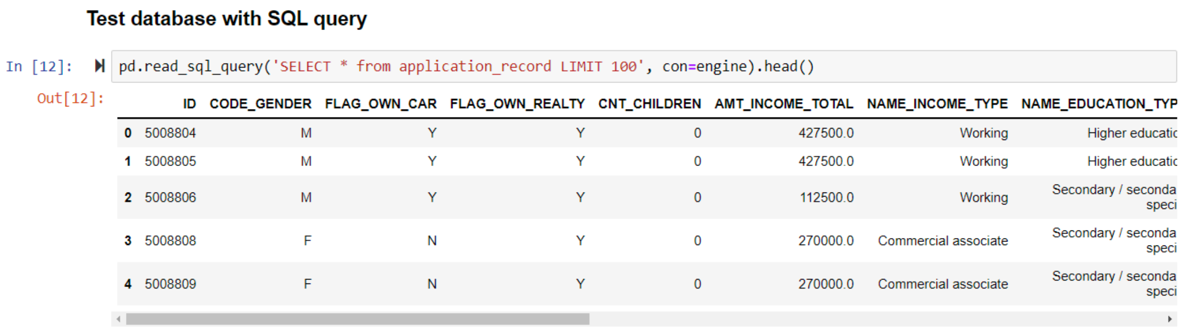
```
In [5]:  def subtract_days(num_days):
             # Get the current date
             today = datetime.now()

             # Always create a timedelta object with a positive number of days
             num_days_ago = timedelta(days=abs(num_days))

             # Subtract the timedelta from the current date to get the date num_days ago
             num_days_ago_date = today - num_days_ago

             # Return the date num_days ago as a string in the desired format
             return num_days_ago_date.strftime("%Y-%m-%d")

         # Use the apply() method to apply the subtract_days() function to every value in the 'days_to_subtract' column
         application_record_df['DAYS_BIRTH'] = application_record_df['DAYS_BIRTH'].apply(subtract_days)

         # Print the resulting DataFrame
         application_record_df.head()
```

Figure 3 - Extracting CSV file into a Pandas DataFrame

4. Load

After we had cleaned and transformed the data into a suitable DataFrame, we loaded the data back into our Postgres database. This was achieved by creating a connection to the database port, which we then tested using the inspector to confirm the table names in the database.

After loading the two DataFrames, we then ran some SQL queries using the SQLAlchemy python library to confirm the data had been loaded correctly into the database. Figure 4 below shows the first query.



Test database with SQL query

```
In [12]:  pd.read_sql_query('SELECT * from application_record LIMIT 100', con=engine).head()
Out[12]:
```

	ID	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALTY	CNT_CHILDREN	AMT_INCOME_TOTAL	NAME_INCOME_TYPE	NAME_EDUCATION_TYP
0	5008804	M	Y	Y	0	427500.0	Working	Higher educatic
1	5008805	M	Y	Y	0	427500.0	Working	Higher educatic
2	5008806	M	Y	Y	0	112500.0	Working	Secondary / seconda speci
3	5008808	F	N	Y	0	270000.0	Commercial associate	Secondary / seconda speci
4	5008809	F	N	Y	0	270000.0	Commercial associate	Secondary / seconda speci

Figure 4 - SQL Query

5. Conclusion

In conclusion, the data analytics ETL project was a success in performing the basic ETL steps. By using a combination of SQL, Python, and ETL frameworks, we were able to extract and transform data, create a database, ensuring the accuracy and integrity of the resulting dataset.

Throughout the project, we worked closely as a team to design and implement the ETL process, using data analytics techniques we have learnt to form these steps as efficiently as possible.

We encountered several challenges and issues during the project, including data quality issues and difficulties in accessing the best data source for the project. However, by collaborating closely and utilizing our technical skills and knowledge, we were able to overcome these obstacles.

Overall, the data analytics ETL project was a success and we believe that the database created will provide an excellent source for future analysis. We look forward to continuing to work on similar projects in the future and using data analysis to provide valuable insights.

References

宋骁 Seanny (2019). Credit Card Approval Prediction. In Kaggle [dataset].
https://www.kaggle.com/datasets/rikdifos/credit-card-approval-prediction?select=credit_record.csv

saurav_12suman (2022). Credit_Card_Approval. In Kaggle [dataset].
https://www.kaggle.com/code/saurav12suman/credit-card-approval