Week 13 - ETL Project Report

Client engagement

As a small data analytics firm, our client approached us with an urgent project to collect information about this year's LEGO sets. They are part of the manufacturing process of the LEGO sets and want to do predictive data modelling to find out what materials will be more in demand next year, based on this year's information.
They have requested the information be put into database format so that it can be accessed in a way that they are already familiar with.

Team selection

Based on the timeframe of 1 week, we have assigned the following team members to this project:

- John Bingley
- Daniel Da Sobral
- Cicily George
- Sylvia Broadbent

Required information

Following our research into LEGO sets, the following information is required to be able to do the predictive data modelling required by our client:

- Popularity – ratings
- Lego sets – names and themes
- Parts – quantity and colour

We have found that ratings are based on the LEGO sets. It is important to know what sets are popular to predict future sales. It is also important to look if certain themes are more popular than others in case new sets are released in the same theme.
Then it is vital that the number, type and colour of all parts can be extracted.

Data Sources (EXTRACT)

Initially we were going to use the LEGO dataset from Kaggle as the base set (https://www.kaggle.com/hapahacks1/lego-database-2020). It has all the LEGO sets from 2020. As this information was taken from the Rebrickable website, we decided to use an API to extract this information ourselves.
Upon further investigation we found out that Rebrickable (https://rebrickable.com/home/) throttles API requests and asks that you download the information instead. So that is what we did.

The information was downloaded as .csv files. This is the simple version of the ERD (for the full ERD, please refer to the Appendix):

- Inventories
  - Inventory parts
    - parts
    - colours
- inventory minifigs
  - minifigs
- inventory sets
  - sets
  - themes

As there is no information from customers on the Rebrickable website we looked elsewhere for information about the ratings of the LEGO sets.
We found this information on the Brickset website (https://brickset.com/) and used an API to extract the information formatted in JSON format.

Data Cleanup (TRANSFORM)

We used the following programs and packages for cleaning up the data:

- Jupyter Notebook
- Python
- Pandas
- API extraction
- JSON file

In Jupyter Notebook using Python with Pandas we loaded the .csv files into dataframes. Then we filtered the information by extracting the information for this year (2020). In some dataframes we also had to change the names of some columns to ensure the connection of the tables are possible in the PostgreSQL database using the primary and foreign keys. Then we dropped the duplicates in the dataframe to ensure a clean set. In the 'colors' dataframe, one column (is_trans) only has the letters t and f. The letters stand for True and False to provide information if the LEGO brick is transparent or not. To make it clear, those letters were changed to True and False.

We used an API query to download the information for each LEGO set in 2020. This was then saved as a JSON file. We used Pandas to read the dictionary inside the JSON file and extracted the 'set number' to ensure we have a joining key and the rating and review count, which is the information we want to add. The information was saved in a table and joined to the 'sets' table.

Type of Database

As the LEGO tables from Rebrickable each have various references to other tables, we have chosen to use a relational database. PostgreSQL is particularly suitable for this type of information as it uses the primary and foreign keys to connect the tables.

pgAdmin & PostgeSQL (LOAD)

As we are using a relational database it is important to verify that the relationships work and that all information in the database is able to be identified and extracted.
To ensure we are presenting a working database to the client we first used an Entity Relationship Diagram (ERD) to layout the tables and identify the primary and foreign keys.
Once that information was available the tables were loaded into PostgreSQL via pgAdmin.
The information from the Pandas dataframes was then add to the tables in pgAdmin via a direct connection to the database in Jupyter notebook.
We have found that it is important to keep the ERD updated as you change names in tables to ensure the same names are used in the separate tables.
We have also found that if the database exists in pgAdmin, writing the information into the database will also create the tables.

Delivery

After intensive testing the database was presented to the client. The client was very happy with the outcome and took possession of the project.

APPENDIX
Entity Relationship Diagram (ERD)