**PROJECT OUTLINE**

**Building a Relational Database for a Bike shop**

**Project Executive:** [**Gia Bao Nguyen (Data Scientist)**](https://www.linkedin.com/in/gia-bao-nguyen-7b3aaa223/)

**Data Source:** [**Bike Store Relational Database**](https://www.kaggle.com/datasets/dillonmyrick/bike-store-sample-database?select=products.csv)

**I/ DATABASE BUILDING**

**1/ DEFINE THE GOALS**

**-Problem:** Many years ago, Bao is a dedicated bike seller, he opened his first store with a passion to become a millionaire. Because of his dedication, many customers around the world has come and experienced his bikes. At first, everything seem easy for him, however, after opening more than 3 stores and having more than 100 loyal customers, Bao soon realizes that he can not manage these customers just by putting all of their information into a csv file.

* **Then he start building a relational database, with an intention of managing his customer effectively.**
* **With this relational database, Bao also realizes that he can put everything online to create an online bike shopping platform**

**2/ CREATE A DATABASE WITH MYSQL**

A diagram of a product

AI-generated content may be incorrect.

*(Image:Data Model of the Database)*

* Based on the provided data model, we can create the database with available csv files by the following DDL on MySQL (Note: DDL needs to be executed in a specific order to run successfully)
* Step 1: Create tables that do not have foreign key: brands, categories, stores

A computer screen shot of a computer

AI-generated content may be incorrect.

A computer code on a white background

AI-generated content may be incorrect.

A screen shot of a computer

AI-generated content may be incorrect.

* Step 2: Create table which have foreign keys that are depend on the above tables: products

A screen shot of a computer code

AI-generated content may be incorrect.

* Step 3: Create tables that have relationship with stores: staffs

A computer screen shot of a computer code

AI-generated content may be incorrect.

* Step 4: Create table that relates to orders: orders

A computer code with text

AI-generated content may be incorrect.

* Step 5: Create tables to connect order\_items and stocks: order\_items, stocks

A computer code with text

AI-generated content may be incorrect.



* Input the csv files into each table following this order:

+ **brands**  
+**categories**  
+ **stores**  
+**customers**  
+**products**   
+**staffs**   
+**orders**   
+**order\_items**   
+**stocks**

**3/ DATA PIPELINE**

**A diagram of a process

AI-generated content may be incorrect.**

*(Image: Data Pipeline)*

1. **Database Connection Setup**:
   * The application connects to a MySQL database named bike\_shop using the flask\_mysqldb extension.
   * Configuration details such as host, user, password, and port are specified to establish the connection.
2. **Data Retrieval for Display**:
   * The index route fetches product information from the database to display on the homepage.
   * It executes a SQL query that joins multiple tables (products, brands, categories, and stocks) to retrieve product details like name, brand, category, price, and available quantity for a specific store (store\_id = 1).
   * The fetched data is then passed to the index.html template for rendering.
3. **Order Processing Pipeline**:
   * The buy\_product route handles the purchase of a product.
   * It receives product details via a POST request in JSON format.
   * The pipeline performs several steps:
     + **Stock Availability Check**: It checks if the product is in stock by querying the stocks table.
     + **Price Retrieval**: It retrieves the product's price from the products table.
     + **Order Creation**: It inserts a new order into the orders table with a status of 'Pending' and the current timestamp.
     + **Order Item Addition**: It calculates the next item\_id for the new order and inserts the product into the order\_items table.
     + **Stock Update**: It decrements the product quantity in the stocks table to reflect the sale.
   * If any step fails, an error is returned. Otherwise, a success message is sent back.
4. **Data Flow**:
   * The data flows from the client (browser) to the Flask application via HTTP requests.
   * The application processes the data by interacting with the MySQL database, performing CRUD (Create, Read, Update, Delete) operations as needed.
   * The results are then sent back to the client, either as a rendered HTML page or a JSON response.
5. **Error Handling**:
   * When the quantity in stocks table reach 0 but the use still click on buy, an error message will pop up.

**4/ DEPLOY THE DATABASE ON LOCAL SERVER WITH FLASK**

* Using Flask, we can create a website that simulate the operation of an online shopping platform. It can connect to MySQL, retrieve product’s information and update the database accordingly when use click on buy
* This simulation platform can only run on local server.

**II/ MACHINE LEARNING**

**1/ DEFINE THE GOALS**

* **Problem:** Once again, a new problem arises. Bao continuously brought in various bicycle brands, but then realized the seriousness of not understanding customer behaviour. Some bike models were almost impossible to sell, causing him to struggle not knowing how to handle it. Then, Bao suddenly realized that he could leverage information from the database to understand the trends of individual customers.

**=> He decided to categorize the bike brands into 3 segments: high for brands with an average price greater than 1000, mid for brands with an average price between 500-1000, and low for brands with an average price below 500.**

**=> With machine learning, Bao can now predict which brand a customer in his list will buy next. This helps him avoid stocking items indiscriminately and anticipate customer demand.**

**2/ QUERY THE DATABASE TO EXTRACT NEEDED FEATURES**

* Needed features:

**+customer\_id:** Id of each customer

**+total\_spent:** Total amount of a customer

**+latest\_purchased\_brand:** The brand of the bike that the customer bought in their most recent purchase.

**+latest\_brand\_tier:** The tier of the bike brand the customer recently purchased.

**+most\_purchased\_brand:** The brand the customer has bought the most over all purchases.

**+latest\_brand\_name:** The name of the bike brand the customer recently purchased

**+most\_purchased\_brand\_name:** The name of the bike brand the customer has bought the most over all purchases

* Using the [SQL script](../Machine_Learning/Query_DataforML.sql) attached on **the machine learning folder on github,** you can query all these features from the bike\_shop database.

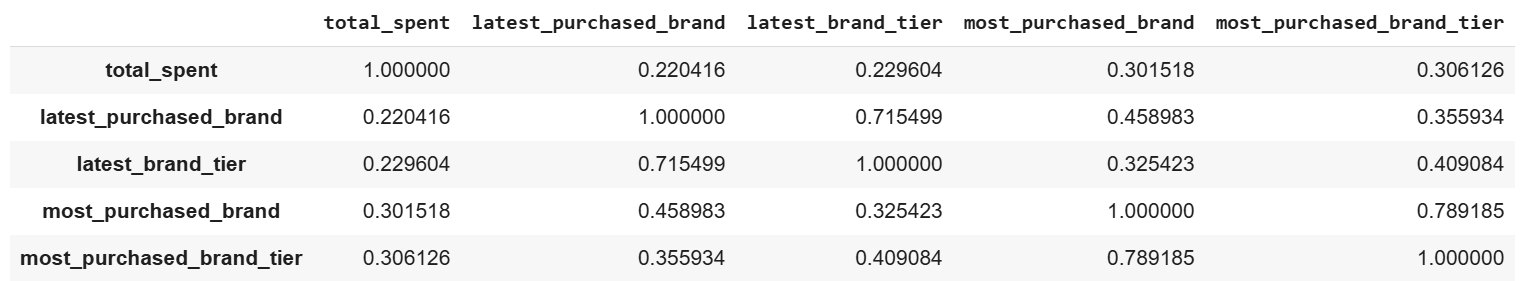
**3/ MODEL BUILDING AND EVALUATION**

* **Define features: latest\_brand\_tier** will be our target features, others will be used as independent features
* **Oversampling:** The data is imbalanced with only 241 tier 1 samples, we will use SMOTE to oversampling class 1 to 1200 samples

A screenshot of a cell phone

AI-generated content may be incorrect.

* **Train-test split:** 30% of the data will be used for testing
* **Algorithm:** we will use logistic regression because our independent features all have a good correlation with the target features (based on the correlation matrix). This algorithm is also very computational effective. Because we have a large dataset that will be updated frequently, using other algorithms such as XGboost or Neural networks may provide more accurate results, but they will cost us a lot of cpu.



* **Model evaluation:**

**A screenshot of a graph

AI-generated content may be incorrect.**

+ The model has an overall accuracy of 81%

+ The model does not perform so good with tier-1 samples, however, its performance on the other class is not that bad. To handle a large amount of data from the database, we prioritize a model that can solve the problem fast.