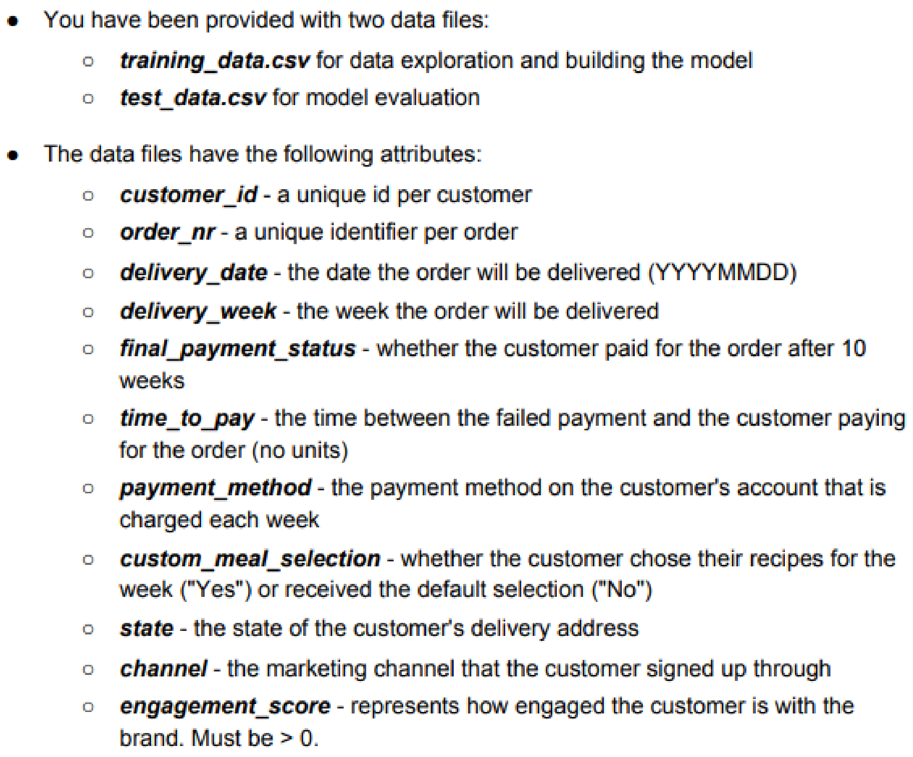
# Machine Learning - Case Study

## Introduction

### Failed Payments

- Company X is a weekly subscription service. Customers have the option to skip a week's delivery or cancel their account; if they do neither they will be charged each week and sent a box of ingredients and recipes.
- Each week, payment can't be taken for a subset of customers (for example, the credit card on the account is expired). We call these "failed payments".
- Customers with a failed payment are contacted, and have the ability to pay for their order in the following days. However, for logistical reasons a decision must be made within a few hours of the missed payment as to whether to ship the unpaid order
- You've come up with the idea to make a machine learning model that informs the logistics team whether to produce and ship each box that missed payment, based on the likelihood of the customer eventually paying for it.

### Data



- You have been provided with two data files:
    - ***training_data.csv*** for data exploration and building the model
    - ***test_data.csv*** for model evaluation
- The data files have the following attributes:
    - ***customer_id*** - a unique id per customer
    - ***order_nr*** - a unique identifier per order
    - ***delivery_date*** - the date the order will be delivered (YYYYMMDD)
    - ***delivery_week*** - the week the order will be delivered
    - ***final_payment_status*** - whether the customer paid for the order after 10 weeks
    - ***time_to_pay*** - the time between the failed payment and the customer paying for the order (no units)
    - ***payment_method*** - the payment method on the customer's account that is charged each week
    - ***custom_meal_selection*** - whether the customer chose their recipes for the week ("Yes") or received the default selection ("No")
    - ***state*** - the state of the customer's delivery address
    - ***channel*** - the marketing channel that the customer signed up through
    - ***engagement_score*** - represents how engaged the customer is with the brand. Must be > 0.

- ***num_prior_orders_failed*** - the number of previous orders the customer had with a failed payment
- ***num_prior_orders_unpaid*** - the number of previous orders the customer had with a failed payment, that they didn't pay as at the time of the current failed order
- ***avg_recipe_rating*** - the average score the customer gave their past recipes

## Part One: Problem Solving

1. The CEO is very interested in your idea to make a model, and would like you to summarise your goal. Write a problem statement: i.e. summarise the challenge you are trying to solve in one sentence/question.

   *Try to make your problem statement SMART - specific, measurable, achievable, realistic and time-bound.*

2. The CEO also wants to know: Do you think shipping orders that fail payments is profitable? Briefly explain your opinion.

# Part Two: SQL

Based on the tables below (filled with example data), write SQL queries to get:

1. A list of unique ***order_number***s that failed payment with delivery dates between 2020-02-08 and 2020-02-12 (inclusive) in Australia

2. For each order that failed payment:
    a. If the order was eventually paid, the number of days between the **first** failed payment and it being paid for by the customer
    b. If the order was never paid, put ***never_paid***

3. ***customer_id***, ***name***, ***phone number*** and ***order_number*** for the most recent order (by delivery date) per customer

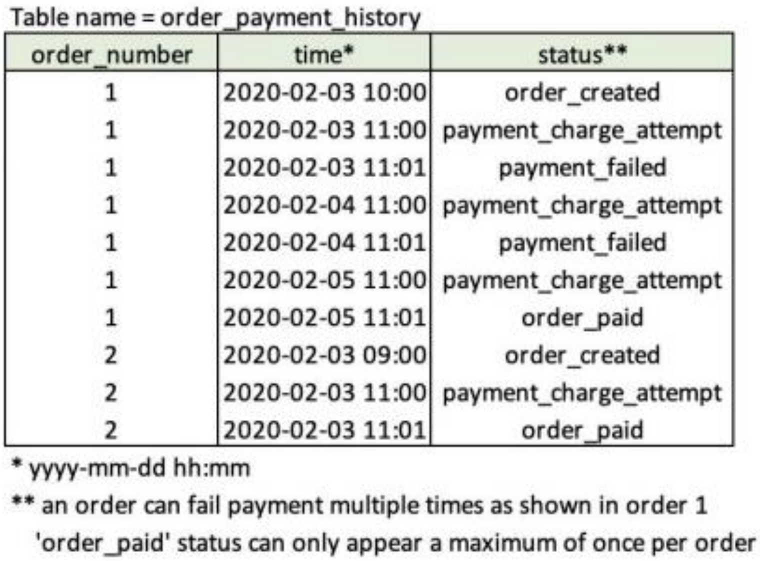


Table name = order_payment_history

| order_number | time* | status** |
|---|---|---|
| 1 | 2020-02-03 10:00 | order_created |
| 1 | 2020-02-03 11:00 | payment_charge_attempt |
| 1 | 2020-02-03 11:01 | payment_failed |
| 1 | 2020-02-04 11:00 | payment_charge_attempt |
| 1 | 2020-02-04 11:01 | payment_failed |
| 1 | 2020-02-05 11:00 | payment_charge_attempt |
| 1 | 2020-02-05 11:01 | order_paid |
| 2 | 2020-02-03 09:00 | order_created |
| 2 | 2020-02-03 11:00 | payment_charge_attempt |
| 2 | 2020-02-03 11:01 | order_paid |

* yyyy-mm-dd hh:mm

** an order can fail payment multiple times as shown in order 1
'order_paid' status can only appear a maximum of once per order

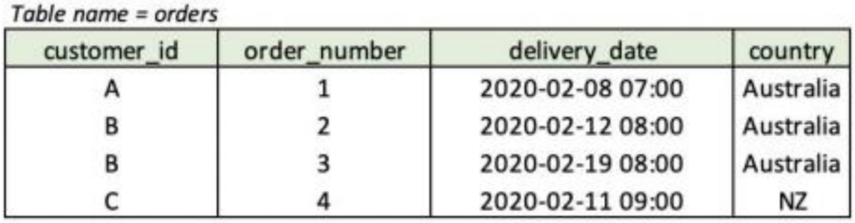


*Table name = orders*

| customer_id | order_number | delivery_date | country |
|---|---|---|---|
| A | 1 | 2020-02-08 07:00 | Australia |
| B | 2 | 2020-02-12 08:00 | Australia |
| B | 3 | 2020-02-19 08:00 | Australia |
| C | 4 | 2020-02-11 09:00 | NZ |

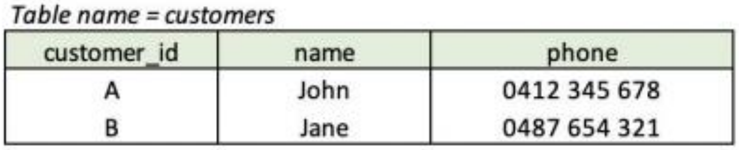


*Table name = customers*

| customer_id | name | phone |
|---|---|---|
| A | John | 0412 345 678 |
| B | Jane | 0487 654 321 |

# Part Three: Python

This section should be completed using Python, preferably with Jupyter notebooks.
Key assessment criteria:

- Code structure and readability
- Code reusability (use of functions)
- Thought process explained through comments or text blocks
- Insights extracted

Using the data from **training_data.csv** write Python code to:

1. Perform data cleaning and check data accuracy. These checks will be applied to the test data in question 5 so use functions to make the code reusable.
    a. Write a function to check data quality
    b. Write a function to clean the data

2. Write functions to build charts and perform EDA (exploratory data analysis). Provide comments for all charts.
    a. Produce one chart to examine the target variable and one to examine ***time_to_pay***.
    b. Produce a chart to examine the relationship between ***engagement_score*** and the target variable (hint: it might be helpful to bucket customers based on the ***engagement_score***).
    c. Produce separate charts to examine the relationship between the target variable and the following features: ***num_prior_orders_unpaid***, ***custom_meal_selection***, ***channel***. The insights might differ between customers with low/high engagement scores - make sure you account for this.

3. Write a function to perform feature engineering:
    a. Create an ***ever_rated_recipe*** binary feature based on the ***avg_recipe_rating*** column
    b. Create an ***is_f*** binary feature based on the ***channel*** (include f1 and f2)

4. Train an ML model to predict whether orders will be paid or unpaid. Use the customer features provided. Note:

- Treat orders with payment status of Cancelled, Refunded or Other as Paid
- Include features that you engineered in question 3.
- Use functions (or other methods of reusable code) to prepare your data for the model.
- Focus on model evaluation. Evaluate model performance and discuss the results.

Using the data from **test_data.csv** write Python code to:

5. Make predictions using the model built in Q4:
    a. Check and clean the data (reuse functions from Q1)
    b. Transform the data (reuse functions from Q4)
    c. Make predictions
    d. Evaluate model performance and discuss the results.