Case Study DS-1: Customer Churn Prediction

Business Context:
The bank is experiencing a higher-than-desired attrition rate in its premium "Gold" checking account segment. The marketing team wants to proactively identify customers at high risk of churning (closing all their accounts) in the next 90 days to target them with retention campaigns.

Data Schema

1. customers: customer_id, age, income_bracket, is_active (Boolean)
2. accounts: account_id, customer_id, account_type, status (e.g., 'Active', 'Closed'), open_date, close_date
3. transactions: transaction_id, account_id, transaction_date, amount, transaction_type
4. interactions: interaction_id, customer_id, interaction_date, interaction_type (e.g., 'Complaint', 'Service Inquiry')

Tasks:

Part A: SQL & Feature Engineering

1. Define your target variable has_churned (1/0). A customer has "churned" if they have closed all their accounts within a specific historical observation window. *(You will be given a specific cutoff date, e.g., "Consider accounts closed in the last 6 months as churners").*
2. Write an SQL query to create a feature-rich dataset for model training. For each customer at a snapshot date (e.g., 90 days before churn/non-churn event), extract features like:
    - num_accounts
    - avg_account_balance
    - transaction_frequency_30d
    - avg_transaction_amount
    - number_complaints_90d
    - customer_tenure_days
    - has_gold_account (Boolean)

Part B: Python Modeling

1. Using **Python** (Pandas, Scikit-learn), load the dataset you generated.
2. Perform necessary data preprocessing: handle missing values, encode categorical variables.
3. Split the data into training and test sets.
4. Train at least **two** classification models (e.g., Logistic Regression and Random Forest) to predict has_churned.
5. Evaluate the models on the test set using appropriate metrics (Accuracy, Precision, Recall, F1-Score, ROC-AUC). Justify your choice of the primary metric for the business context.

Part C: Presentation

1. Present your findings to a non-technical business audience.
2. Explain which model you would deploy and why.
3. What are the top 3 features driving churn according to your best model? Explain what this tells the business about *why* customers might be leaving.
4. How would the marketing team use your model's output? What are the potential risks or ethical considerations of using such a model?

Case Study DS-2: Transaction Anomaly Detection

Business Context:
The Fraud Detection unit wants to move from a rule-based system to a more adaptive, machine-learning-based approach to identify suspicious transactions in real-time. Your task is to build a proof-of-concept anomaly detection model.

Data Sources (Provided as SQLite Database Tables):

1. customers: customer_id, customer_segment
2. accounts: account_id, customer_id, account_type
3. transactions: transaction_id, account_id, transaction_datetime, amount, merchant_category, transaction_location

(Note: The data will be mostly "normal" transactions, with a very small, known set of labeled fraud transactions for final evaluation.)

Tasks:

Part A: SQL & Feature Engineering (Intermediate)
Write an SQL query to create a transactional dataset suitable for anomaly detection. Create features that capture contextual spending behavior, such as:

- transaction_amount
- time_since_last_transaction
- is_weekend
- is_night (e.g., 8 PM - 6 AM)
- avg_amount_7d (the customer's average transaction amount in the last 7 days)
- count_transactions_7d (the customer's transaction count in the last 7 days)

Part B: Python Modeling (Unsupervised/Semi-Supervised)

1. Using **Python**, perform exploratory data analysis (EDA) to understand the distribution of the amount and other key features.
2. **Option 1 (Unsupervised):** Use an unsupervised learning algorithm like **Isolation Forest** or **Local Outlier Factor (LOF)** on the *unlabeled* dataset to score each transaction's anomaly likelihood.
3. **Option 2 (Semi-Supervised):** If you are given a small set of known fraudulent transactions, use it to create a labeled dataset. Train a model like **XGBoost** on a highly imbalanced classification task, using techniques like SMOTE or class weights.
4. Evaluate your model's performance on the hold-out test set (which contains labeled fraud). Use metrics like Precision-Recall AUC since the data is highly imbalanced.

Part C: Presentation & Operationalization

1. Present your model to the Fraud Detection team.
2. How does your model improve upon a simple rule like "flag all transactions > $10,000"?
3. Explain one key insight your EDA or model provided about the nature of anomalous transactions.
4. What would be the next steps to operationalize this model into a real-time system? Consider the need for feature engineering in a streaming context. What is a major challenge you foresee?