# Take-Home Test for Data Engineer

**Objective:**

Build a data pipeline using Python with Pandas or PySpark or SparkSQL to model a users referral program and implement basic business logic to identify and avoid potential fraud. Ensure the code structure is readable and maintainable. Provide the model documentation to ensure business users understand.

**Requirements:**

1. **Script:** Develop a script to process and analyze the referral data.
2. **Docker:** Containerize the application using Docker.
3. **Documentation:** Provide documentation including data dictionary and to operate the docker/setup the environment.

## Business Flow for Referral Program

**Overview**

The referral program aims to incentivize existing users (referrers) to bring new users (referees) to the platform. The program tracks referrals, assigns rewards, and verifies transactions to ensure the integrity of the program. The flow includes the interaction of several tables:

- **lead_logs**
- **user_referrals**
- **user_referral_logs**
- **user_logs**
- **user_referral_statuses**
- **referral_rewards**
- **Paid_transactions**

### Source Referral

1. **At Web Landing Page Free Trial:**
    - When a user fills out a form as a lead, it gets recorded as a referral source (online).
    - Referrers could share information/contact about referees at the club (offline), and input to our system by Customer Service.
2. **At App Sign Up Flow:**
    - During the sign-up process in application, new users can be referred by existing users, creating a referral record.
3. **Offline from Walk-ins:**

- Users who walk into the club, give a referrer code and create transactions are recorded as offline referrals.

**Detailed Business Flow**

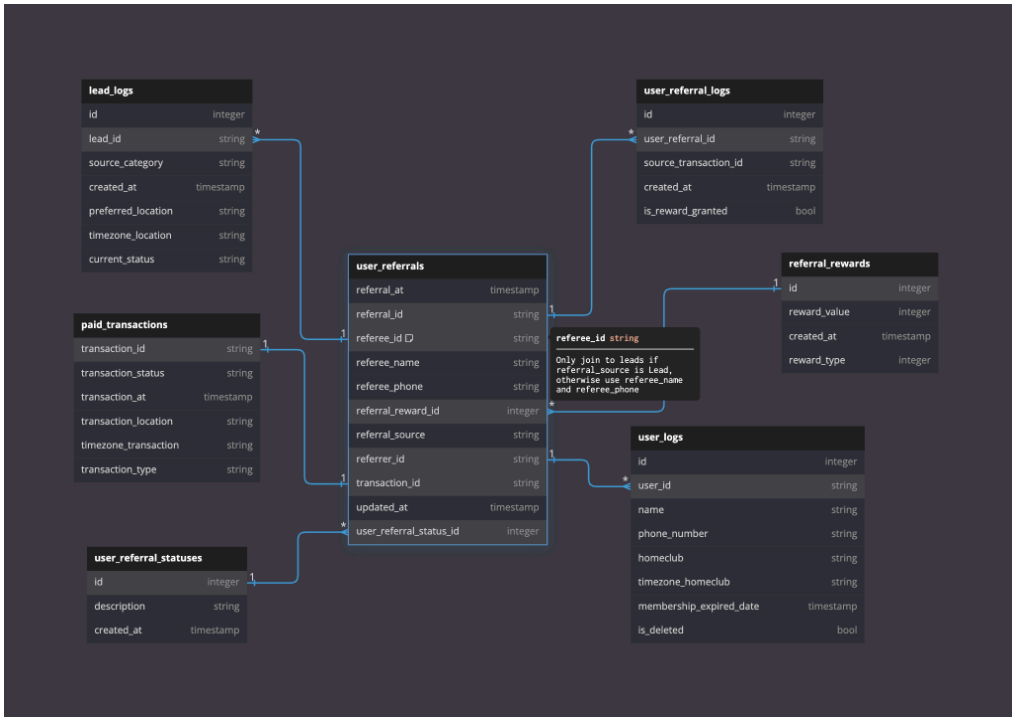
1. **User Registration and Referral Submission**
   - A new user (referee) signs up using a referral code provided by an existing user (referrer).
   - A record is created in user_referrals to track this referral.
   - **user_referrals Table:**
     i. referral_id: Unique identifier for the referral.
     ii. referral_at: Timestamp when the referral was created.
     iii. referrer_id: ID of the user who referred the new user.
     iv. referee_id: ID of the new user (only used if referral_source is "Lead").
     v. referee_name: Name of the new user.
     vi. referee_phone: Phone number of the new user.
     vii. referral_source: Source of the referral (e.g., "User Sign Up", "Draft Transaction", "Lead").
     viii. referral_reward_id: ID of the referral reward.
     ix. transaction_id: ID of the transaction linked to the referral.
     x. updated_at: Timestamp when the referral was last updated.
     xi. user_referral_status_id: ID of the referral status.
2. **Referral Details**
   - Additional details about the referral, such as the transaction ID linked to the referral and reward status, are recorded in user_referral_logs.
   - **user_referral_logs Table:**
     i. id: Unique log id.
     ii. user_referral_id: Foreign key linking to user_referrals.
     iii. source_transaction_id: ID of the transaction that resulted from the referral.
     iv. created_at: Timestamp when the changes are created.
     v. is_reward_granted: Boolean indicating if the reward has been granted.
3. **User Information**
   - Information about users (both referrers and referees) is stored in the user_logs table.
   - **user_logs Table:**
     i. id: Unique log id.
     ii. user_id: ID of the user.
     iii. name: Name of the user.
     iv. phone_number: Phone number of the user.
     v. homeclub: Home club of the user.
     vi. membership_expired_date: Date when the user's membership expires.
     vii. is_deleted: Flag indicating if the user's account is deleted.
4. **Referral Status Tracking**

- The status of each referral is tracked using user_referral_statuses.
- **user_referral_statuses Table:**
    i. id: Unique identifier for the referral status.
    ii. description: Description of the status (e.g., "Berhasil", "Menunggu", "Tidak Berhasil").
    iii. created_at: Timestamp when the status was created.

5. **Reward Management**
    - Rewards for successful referrals are managed in referral_rewards.
    - **referral_rewards Table:**
        i. id: Unique identifier for the reward.
        ii. reward_value: Value of the reward.
        iii. reward_type: Type of reward (e.g., monetary, points).
        iv. created_at: Timestamp when the reward was created.
6. **Transaction Verification**
    - Transactions resulting from referrals are recorded in paid_transactions.
    - **paid_transactions Table:**
        i. transaction_id: Unique ID of the transaction.
        ii. transaction_status: Status of the transaction (e.g., "PAID").
        iii. transaction_at: Timestamp when the transaction occurred.
        iv. transaction_location: Location of the transaction.
        v. timezone_transaction: Timezone of the transaction.
        vi. transaction_type: Type of transaction.
7. **Lead Information**
    - Information about leads is stored in lead_logs.
    - **lead_logs Table:**
        i. id: Unique id log.
        ii. lead_id: ID of the lead.
        iii. source_category: Category of the lead source.
        iv. created_at: Timestamp when the lead was created.
        v. preferred_location: Preferred location of the lead.
        vi. timezone_location: Timezone of the preferred location.
        vii. current_status: Current status of the lead.

## Table Structure

Reference ERD: https://dbdiagram.io/d/take-home-test-6694f17a9939893daef1260f




## Test Instructions

### 1. Script Development:

**Task:**

1. **Data Loading:**
    - Load all CSV files into DataFrames.
2. **Data Cleaning:**
    - Clean the data by handling missing values, correcting data types, and removing duplicates.
3. **Data Processing:**
    - **Time Adjustment**: All timestamp value is in UTC, you need to convert to local time.
    - **Join Tables:** Merge the tables appropriately, make sure there is no duplicate.
    - **Handling Null**: Null value should not exist/removed.
    - **String Adjustment**: Initcap should apply in string value, unless the club name.
    - **Source Category:** Determine referral_source_category using the following logic:

```
%sql
CASE
  WHEN referral_source = 'User Sign Up' THEN 'Online'
  WHEN referral_source = 'Draft Transaction' THEN 'Offline'
  WHEN referral_source = 'Lead' THEN leads.source_category
END AS referral_source_category
```

4. **Basic Business Logic Implementation,** Implement these logics to detect potential fraud in a new column with name is_business_logic_valid (type:bool), such as**:**
    - **Valid Referral Rewards:**
        - **Condition 1:** A referral reward is considered **valid** if:
            1. The reward value is greater than 0.
            2. The referral status is "Berhasil" (Successful).
            3. The referral has a transaction ID.
            4. The transaction status for the referral is "PAID".
            5. The transaction type for the referral is "NEW".
            6. The transaction occurred after the referral was created.
            7. The transaction happened in the same month as the referral creation.
            8. The referrer’s membership has not expired.
            9. The referrer’s account is not deleted.
            10. The reward has been granted to the referee.
        - **Condition 2:** A referral reward is considered **valid** if:
            1. The referral status is either "Menunggu" (Pending) or "Tidak Berhasil" (Failed).
            2. There is no reward value assigned.
    - **Invalid Referral Rewards:**
        - **Condition 1:** A referral reward is considered **invalid** if:
            1. The reward value is greater than 0.
            2. The referral status is not "Berhasil" (Successful).
        - **Condition 2:** A referral reward is considered **invalid** if:
            1. The reward value is greater than 0.
            2. There is no transaction ID for the referral.
        - **Condition 3:** A referral reward is considered **invalid** if:
            1. There is no reward value assigned.
            2. The referral has a transaction ID.
            3. The transaction status for the referral is "PAID".
            4. The transaction occurred after the referral was created.
        - **Condition 4:** A referral reward is considered **invalid** if:
            1. The referral status is "Berhasil" (Successful).

2. The reward value is either null or 0.
- **Condition 5:** A referral reward is considered **invalid** if:
    1. The transaction occurred before the referral was created.
- If you found any invalid business logic is a plus.

5. **Output:**
    - Uploading the report to the cloud (eg. Google Drive / Google Sheet / Dropbox / BigQuery / Google Cloud Storage) is a plus. Putting any credentials in the script is prohibited.
    - Generate a report/table (in CSV format) that indicates whether each referral reward is valid or invalid based on the business logic.
    - The final report includes the following columns and sample data types:

| **Column Name** | **Data Type** | **Sample Value** |
|---|---|---|
| referral_details_id | INTEGER | 101 |
| referral_id | STRING | "9331c8f144dad5a3b8e4a10467b4343a" |
| referral_source | STRING | "User Sign Up" |
| referral_source_category | STRING | "Online" |
| referral_at | DATETIME | "2023-06-15 14:35:00" |
| referrer_id | STRING | "2c71c5d66c7e12a0b3c200ba6ed3b78e" |
| referrer_name | STRING | "John Doe" |
| referrer_phone_number | STRING | "123-456-7890" |
| referrer_homeclub | STRING | "PERMATA HIJAU" |
| referee_id | STRING | "f1327c9d6d4efee6ad69e7e467b605b9" |
| referee_name | STRING | "Jane Smith" |
| referee_phone | STRING | "987-654-3210" |
| referral_status | STRING | "Berhasil" |
| num_reward_days | INT | 30 |
| transaction_id | STRING | "1d1eb8a9e864a1cccb2d850398461807" |
| transaction_status | STRING | "Paid" |
| transaction_at | DATETIME | "2023-06-20 10:00:00" |
| transaction_location | STRING | "BENHIL" |
| transaction_type | STRING | "New" |
| updated_at | DATETIME | "2023-06-21 12:00:00" |
| reward_granted_at | DATETIME | "2023-06-30 14:00:00" |
| is_business_logic_valid | BOOLEAN | TRUE |

**2. Docker Containerization:**

1. **Dockerfile:**
    - Create a Dockerfile to containerize the application.
    - Ensure that all dependencies are installed, including Python, Pandas, PySpark, and any other required libraries.
2. **Running Instructions:**
    - Provide clear instructions on how to build and run the Docker container, and export the report file.
    - The report should be stored outside the container.

**3. Documentation:**

1. **Data Dictionary:**
    - Provide a data dictionary that defines each column in the output report, including the meaning/descriptions, data type, and any constraints or relevant notes.
    - Include a sheet file that provides this information.
2. **Script Documentation:**
    - Comment the script thoroughly to explain each step and the logic behind it.
    - Include a README file that provides an overview of the project, setup instructions, and usage details.
    - If you upload the report to cloud storage, please provide the instructions to set up the credentials. Putting any credentials in the script is prohibited.

## Submission Instructions

1. **Script:** The Python script file (your_script.py) should exist.
2. **Docker:** The Dockerfile should exist.
3. **Documentation:** The data dictionary in Excel format, and README file to run the code.
4. **Repository Submission**: Push all assets to your designated repository. Ensure the repository includes the Python script, Dockerfile, data dictionary, README file, and any additional documentation or configurations required.
5. **To complete the submission process**, you will need to email the link to your repository to the Talent Acquisition Team.

**Example Skeleton of the Python Script** (You can create it in your own style)

```
% python
# [Import library]

# Initialize Spark session

# Load data
# [Implement load data here]

# Data cleaning
# [Implement data cleaning logic here]

# Data processing
# [Implement data processing and feature engineering logic here]

# Basic business logic implementation to detect fraud
# [Implement fraud detection logic here]

# Output
# [Generate and save the report (uploading a file is a plus)]

# Stop Spark session
```

**Example Skeleton of the Dockerfile** (You can create it in your own style)

```
% Dockerfile

# Use official Python image

# Set working directory

# Copy requirements file

# Install dependencies

# Copy application code

# Command to run the application
```