Part 1: Data exploration

1. **Setup git cloning**

```
! git clone https://github.com/anushagj/friend-up-your-cash-app-game.git
! pip install prefect==1.0 -U
```

```
    fatal: destination path 'friend-up-your-cash-app-game' already exists and is not an empty directory.
    Requirement already satisfied: prefect==1.0 in /usr/local/lib/python3.10/dist-packages (1.0.0)
    Requirement already satisfied: click>=7.0 in /usr/local/lib/python3.10/dist-packages (from prefect==1.0) (8.1.7)
    Requirement already satisfied: cloudpickle>=1.3.0 in /usr/local/lib/python3.10/dist-packages (from prefect==1.0) (2.2.1)
    Requirement already satisfied: croniter>=0.3.24 in /usr/local/lib/python3.10/dist-packages (from prefect==1.0) (1.4.1)
    Requirement already satisfied: dask>=2021.06.0 in /usr/local/lib/python3.10/dist-packages (from prefect==1.0) (2023.8.1)
    Requirement already satisfied: distributed>=2.17.0 in /usr/local/lib/python3.10/dist-packages (from prefect==1.0) (2023.8.1)
    Requirement already satisfied: docker>=3.4.1 in /usr/local/lib/python3.10/dist-packages (from prefect==1.0) (6.1.3)
    Requirement already satisfied: importlib-resources>=3.0.0 in /usr/local/lib/python3.10/dist-packages (from prefect==1.0) (6.
    Requirement already satisfied: marshmallow>=3.0.0b19 in /usr/local/lib/python3.10/dist-packages (from prefect==1.0) (3.20.1)
    Requirement already satisfied: marshmallow-oneofschema>=2.0.0b2 in /usr/local/lib/python3.10/dist-packages (from prefect==1.
    Requirement already satisfied: msgpack>=0.6.0 in /usr/local/lib/python3.10/dist-packages (from prefect==1.0) (1.0.5)
    Requirement already satisfied: mypy-extensions>=0.4.0 in /usr/local/lib/python3.10/dist-packages (from prefect==1.0) (1.0.0)
    Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from prefect==1.0) (23.1)
    Requirement already satisfied: pendulum>=2.0.4 in /usr/local/lib/python3.10/dist-packages (from prefect==1.0) (2.1.2)
    Requirement already satisfied: python-dateutil>=2.7.0 in /usr/local/lib/python3.10/dist-packages (from prefect==1.0) (2.8.2)
    Requirement already satisfied: pyyaml>=3.13 in /usr/local/lib/python3.10/dist-packages (from prefect==1.0) (6.0.1)
    Requirement already satisfied: python-box>=5.1.0 in /usr/local/lib/python3.10/dist-packages (from prefect==1.0) (7.1.1)
    Requirement already satisfied: python-slugify>=1.2.6 in /usr/local/lib/python3.10/dist-packages (from prefect==1.0) (8.0.1)
    Requirement already satisfied: pytz>=2018.7 in /usr/local/lib/python3.10/dist-packages (from prefect==1.0) (2023.3.post1)
    Requirement already satisfied: requests>=2.25 in /usr/local/lib/python3.10/dist-packages (from prefect==1.0) (2.31.0)
    Requirement already satisfied: tabulate>=0.8.0 in /usr/local/lib/python3.10/dist-packages (from prefect==1.0) (0.9.0)
    Requirement already satisfied: toml>=0.9.4 in /usr/local/lib/python3.10/dist-packages (from prefect==1.0) (0.10.2)
    Requirement already satisfied: urllib3>=1.26.0 in /usr/local/lib/python3.10/dist-packages (from prefect==1.0) (2.0.4)
    Requirement already satisfied: fsspec>=2021.09.0 in /usr/local/lib/python3.10/dist-packages (from dask>=2021.06.0->prefect==
    Requirement already satisfied: partd>=1.2.0 in /usr/local/lib/python3.10/dist-packages (from dask>=2021.06.0->prefect==1.0)
    Requirement already satisfied: toolz>=0.10.0 in /usr/local/lib/python3.10/dist-packages (from dask>=2021.06.0->prefect==1.0)
    Requirement already satisfied: importlib-metadata>=4.13.0 in /usr/local/lib/python3.10/dist-packages (from dask>=2021.06.0->
    Requirement already satisfied: jinja2>=2.10.3 in /usr/local/lib/python3.10/dist-packages (from distributed>=2.17.0->prefect=
    Requirement already satisfied: locket>=1.0.0 in /usr/local/lib/python3.10/dist-packages (from distributed>=2.17.0->prefect==
    Requirement already satisfied: psutil>=5.7.2 in /usr/local/lib/python3.10/dist-packages (from distributed>=2.17.0->prefect==
    Requirement already satisfied: sortedcontainers>=2.0.5 in /usr/local/lib/python3.10/dist-packages (from distributed>=2.17.0-
    Requirement already satisfied: tblib>=1.6.0 in /usr/local/lib/python3.10/dist-packages (from distributed>=2.17.0->prefect==1
    Requirement already satisfied: tornado>=6.0.4 in /usr/local/lib/python3.10/dist-packages (from distributed>=2.17.0->prefect=
    Requirement already satisfied: zict>=2.2.0 in /usr/local/lib/python3.10/dist-packages (from distributed>=2.17.0->prefect==1.
    Requirement already satisfied: websocket-client>=0.32.0 in /usr/local/lib/python3.10/dist-packages (from docker>=3.4.1->pref
    Requirement already satisfied: pytzdata>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pendulum>=2.0.4->prefect==1
    Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.7.0->prefect==1.
    Requirement already satisfied: text-unidecode>=1.3 in /usr/local/lib/python3.10/dist-packages (from python-slugify>=1.2.6->p
    Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests>=2.25->pre
    Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests>=2.25->prefect==1.0) (
    Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests>=2.25->prefect==
    Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.10/dist-packages (from importlib-metadata>=4.13.0->dask>=
    Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.10/dist-packages (from jinja2>=2.10.3->distributed>
```

2. Let's quickly explore the data

```
# Load the sample data into a dataframe. A dataframe is a data structure that organizes data into a 2-dimensional table of rows a

import pandas as pd
parquet_file_path = '/content/friend-up-your-cash-app-game/Dataset/cash_friends.parquet'
cash_friends = pd.read_parquet(parquet_file_path)
cash_friends.head()
```

	user_id	account_creation_date	gender	count_num_transactions_last_yr	sum_amount_spent_all_time_usd	current_cash_acco
0	LyuLjUo0dH	2020-04-01	Male	14	1383.0	
1	86lAOsc1Gh	2015-07-19	Male	15	528.0	
2	Ycl21zkiL1	2019-04-23	Female	16	720.0	
3	10zlKlUH4r	2018-11-29	Male	30	1062.0	
4	dflMuC8Yz8	2015-10-06	Male	11	199.0	

3. **Create Prefect Task**

```
import prefect
from prefect import task, Flow
@task
def hello_task():
  logger = prefect.context.get("logger")
  logger.info("Hello world!")

flow = Flow("hello-flow", tasks=[hello_task])
flow.run()
```

```
[2023-09-25 02:40:15+0000] INFO - prefect.FlowRunner | Beginning Flow run for 'hello-flow'
INFO:prefect.FlowRunner:Beginning Flow run for 'hello-flow'
[2023-09-25 02:40:15+0000] INFO - prefect.TaskRunner | Task 'hello_task': Starting task run...
INFO:prefect.TaskRunner:Task 'hello_task': Starting task run...
[2023-09-25 02:40:15+0000] INFO - prefect.hello_task | Hello world!
INFO:prefect.hello_task:Hello world!
[2023-09-25 02:40:15+0000] INFO - prefect.TaskRunner | Task 'hello_task': Finished task run for task with final state: 'Succ
INFO:prefect.TaskRunner:Task 'hello_task': Finished task run for task with final state: 'Success'
[2023-09-25 02:40:15+0000] INFO - prefect.FlowRunner | Flow run SUCCESS: all reference tasks succeeded
INFO:prefect.FlowRunner:Flow run SUCCESS: all reference tasks succeeded
<Success: "All reference tasks succeeded.">
```

4. [For personal exploration] Connecting local prefect to our cloud prefect with the Python SDK

Create a Free-Tier Prefect Account

1. In a new tab, go to https://cloud.prefect.io/
2. Click Sign in with Google option and use the new google account created in the previous step.
3. Click Next, then click TO THE DASHBOARD

Create an API key : https://cloud.prefect.io/user/keys, **save the key**!

```
flow.register(project_name="cash_find_friends")
```

```
! prefect auth login --key <Your KEY>
```

```
! prefect create project cash_find_friends
```

```
! prefect agent local start
```

Next we follow the link that was generated and select quick run and we will see our flow run in the cloud !

The above task in Prefect Cloud

5. Create a Free Google Account

Create a google account here (if you don't already have one)

Create a Free Google Cloud Platform Account

In a new tab go to https://console.cloud.google.com/. Then in the top left, click on Select a **project > new project**

6. Create a Table in Big Query using Prefect

```
import os
from google.cloud import bigquery
from prefect import task, Flow, Parameter
import pandas as pd

#TO BE UPDATED BY YOU
```

```
PROJECT_ID = "cash-friends-399817"
DATASET_NAME = "Friends"
TABLE_NAME = "cash_friends"

#TO BE UPDATED BY YOU
os.environ["GOOGLE_APPLICATION_CREDENTIALS"] = "/content/cash-friends-399817-9e32c8d023fd.json"

# Function to create a new table in BigQuery
@task
def create_table(project_id, dataset_name, table_name):
  client = bigquery.Client(project=project_id)

  # Define the schema for your table (change the fields accordingly)
  schema = [
    bigquery.SchemaField("user_id", "STRING"),
    bigquery.SchemaField("account_creation_date", "STRING"),
    bigquery.SchemaField("gender", "STRING"),
    bigquery.SchemaField("count_num_transactions_last_yr", "INTEGER"),
    bigquery.SchemaField("sum_amount_spent_all_time_usd", "FLOAT"),
    bigquery.SchemaField("current_cash_account_balance_usd", "FLOAT"),
    bigquery.SchemaField("current_bitcoin_account_balance_btc", "FLOAT"),
    bigquery.SchemaField("current_stock_account_balance_usd", "FLOAT"),
    bigquery.SchemaField("cash_card_enabled", "STRING"),
    bigquery.SchemaField("direct_deposit_enabled", "STRING"),
    bigquery.SchemaField("cash_boost_used", "STRING"),
    bigquery.SchemaField("most_interacted_user_index", "INTEGER"),
    bigquery.SchemaField("user_occupation", "STRING"),
    bigquery.SchemaField("location", "STRING"),
    bigquery.SchemaField("most_used_cash_app_feature", "STRING"),
    bigquery.SchemaField("account_age_yr","INTEGER"),
    bigquery.SchemaField("most_interacted_user_id","STRING")
  ]

  table_ref = client.dataset(dataset_name).table(table_name)
  table = bigquery.Table(table_ref, schema=schema)

  # Create the table
  table = client.create_table(table)
  print(f"Table {table.project}.{table.dataset_id}.{table.table_id} created.")
```

7. Upload data from the parquet file into BigQuery

```
# Function to upload Parquet data to BigQuery table
@task
def upload_parquet_to_bigquery(parquet_file_path, project_id, dataset_name, table_name):
  df = pd.read_parquet(parquet_file_path)

  df['account_creation_date'] = df['account_creation_date'].dt.strftime('%Y-%m-%d %H:%M:%S')


  # Initialize a BigQuery client
  client = bigquery.Client()


  # Define the job configuration
  job_config = bigquery.LoadJobConfig()
  job_config.source_format = bigquery.SourceFormat.PARQUET
  job_config.autodetect = True  # Automatically detect schema

  # Upload the DataFrame to BigQuery
  table_ref = client.dataset(dataset_name).table(table_name)
  job = client.load_table_from_dataframe(df, table_ref, job_config=job_config)

  # Wait for the job to complete
  job.result()

  print(f"Loaded {job.output_rows} rows into {dataset_name}:{table_name}")


with Flow("Parquet to BigQuery Flow") as flow:
    # Create the BigQuery table
    create_table_task = create_table(PROJECT_ID, DATASET_NAME, TABLE_NAME)

    # Upload Parquet data to the table
    upload_parquet_task = upload_parquet_to_bigquery(parquet_file_path, PROJECT_ID, DATASET_NAME, TABLE_NAME)
```

```
flow.run()
```

```
[2023-09-25 02:44:31+0000] INFO - prefect.FlowRunner | Beginning Flow run for 'Parquet to BigQuery Flow'
INFO:prefect.FlowRunner:Beginning Flow run for 'Parquet to BigQuery Flow'
[2023-09-25 02:44:31+0000] INFO - prefect.TaskRunner | Task 'create_table': Starting task run...
INFO:prefect.TaskRunner:Task 'create_table': Starting task run...
Table cash-friends-399817.Friends.cash_friends created.
[2023-09-25 02:44:32+0000] INFO - prefect.TaskRunner | Task 'create_table': Finished task run for task with final state: 'Su
INFO:prefect.TaskRunner:Task 'create_table': Finished task run for task with final state: 'Success'
[2023-09-25 02:44:32+0000] INFO - prefect.TaskRunner | Task 'upload_parquet_to_bigquery': Starting task run...
INFO:prefect.TaskRunner:Task 'upload_parquet_to_bigquery': Starting task run...
Loaded 5000 rows into Friends:cash_friends
[2023-09-25 02:44:36+0000] INFO - prefect.TaskRunner | Task 'upload_parquet_to_bigquery': Finished task run for task with fi
INFO:prefect.TaskRunner:Task 'upload_parquet_to_bigquery': Finished task run for task with final state: 'Success'
[2023-09-25 02:44:36+0000] INFO - prefect.FlowRunner | Flow run SUCCESS: all reference tasks succeeded
INFO:prefect.FlowRunner:Flow run SUCCESS: all reference tasks succeeded
<Success: "All reference tasks succeeded.">
```

Part 2: Encoding & Embeddings

8. Setup

```
import pandas as pd
from sklearn import preprocessing
from scipy.spatial import distance
```

9. Encode Cash Friends Categorical Features

```
categorical_cols = ["user_occupation", "most_used_cash_app_feature", "gender"]
binary_cols = ["cash_card_enabled", "direct_deposit_enabled", "cash_boost_used", ]


# Encode the categorical columns

# use built in encoder preprocessing.LabelEncoder()
def encode_categorical_columns(cols, cash_friends):
    categorical_encoders = {}
    for col in cols:
        # create new label encoder for this column
        label_encoder = preprocessing.LabelEncoder()
        # Fit label encoder to the column values and return encoded labels.
        encoded_col = label_encoder.fit_transform(cash_friends[col].values.tolist())
        # save encoded column values in new column
        cash_friends[col + "_encoded"] = encoded_col
        # save encoder for this column
        categorical_encoders[col] = label_encoder
    return cash_friends, categorical_encoders


# Encode the binary columns

# use built in encoder preprocessing.LabelBinarizer()
def encode_binary_columns(cols, cash_friends):
    binary_encoders = {}
    for col in cols:
        label_encoder = preprocessing.LabelBinarizer()
        encoded_col = label_encoder.fit_transform(cash_friends[col].values.tolist())
        cash_friends[col + "_encoded"] = encoded_col
        binary_encoders[col] = label_encoder
    return cash_friends, binary_encoders


# Encode the columns
cash_friends, categorical_encoders = encode_categorical_columns(categorical_cols, cash_friends)
cash_friends, binary_encoders = encode_binary_columns(binary_cols, cash_friends)
```

10. Drop all original columns categorical & binary columns

```
# Drop non numerical columns for distance calculation
vector_df = cash_friends.drop(columns=['user_id', 'most_interacted_user_id', 'account_creation_date', 'gender', 'cash_card_enable
       'most_used_cash_app_feature'])
```

11. Compute Vector Distances

```
# use scipy distance functions
# manhattan : distance.cityblock
# euclidean : distance.euclidean

def manhattan_distance(vector_1, vector_2):
    return distance.cityblock(vector_1, vector_2)

def euclidean_distance(vector_1, vector_2):
    return distance.euclidean(vector_1, vector_2)
```

12. Lets get the top 3 recommended friends for user 0

```
# Using row 0 as our target row
target_row = vector_df.iloc[0]
```

```
# Compute vector distances
manhatten_distances = vector_df.apply(lambda row: manhattan_distance(target_row, row), axis=1)
euclidian_distances = vector_df.apply(lambda row: euclidean_distance(target_row, row), axis=1)
vector_df["manhattan_distances"] = manhatten_distances
vector_df["euclidian_distances"] = euclidian_distances
```

13. Rank the other users and get the top 3 recommended for each distance metric

```
euclidian_distances = vector_df["euclidian_distances"]
euc_dict = euclidian_distances.to_dict()
ordered_customers_euc =[(customer, distance) for customer, distance in euc_dict.items()]
ordered_customers_euc.sort(key=lambda elem: elem[1])
ordered_customers_euc[:4]
```

```
[(0, 0.0),
 (1772, 206.0826360953295),
 (981, 280.55483314318434),
 (2443, 300.237612733648)]
```

```
manhattan_distances = vector_df["manhattan_distances"]
man_dict = manhattan_distances.to_dict()
ordered_customers_man =[(customer, distance) for customer, distance in man_dict.items()]
ordered_customers_man.sort(key=lambda elem: elem[1])
ordered_customers_man[:4]
```

```
[(0, 0.0), (1772, 304.23), (1183, 499.1), (3320, 526.31)]
```

14. Compare target user to recommended users

```
target_user = cash_friends.iloc[0]
target_user
```

```
user_id                                      LyuLjUo0dH
account_creation_date               2020-04-01 00:00:00
gender                                             Male
count_num_transactions_last_yr                       14
sum_amount_spent_all_time_usd                    1383.0
current_cash_account_balance_usd                  714.0
current_bitcoin_account_balance_btc                2.27
current_stock_account_balance_usd                1432.0
cash_card_enabled                                   Yes
direct_deposit_enabled                              Yes
cash_boost_used                                     Yes
most_interacted_user_index                          442
user_occupation                                  Lawyer
```

```
    location                                       Wyoming
    most_used_cash_app_feature          Peer to Peer Payment
    account_age_yr                                       3
    most_interacted_user_id                     dt8BG7TNjO
    user_occupation_encoded                              6
    most_used_cash_app_feature_encoded                   4
    gender_encoded                                       1
    cash_card_enabled_encoded                            1
    direct_deposit_enabled_encoded                       1
    cash_boost_used_encoded                              1
    Name: 0, dtype: object
```

```
# Check recommender user using Euclidean distance
```

```
recommender_user_id = ordered_customers_euc[1][0]
recommended_user = cash_friends.iloc[recommender_user_id]
recommended_user
```

```
    user_id                                      FeKVVsuTml
    account_creation_date               2020-06-16 00:00:00
    gender                                           Female
    count_num_transactions_last_yr                       16
    sum_amount_spent_all_time_usd                    1377.0
    current_cash_account_balance_usd                  698.0
    current_bitcoin_account_balance_btc                2.04
    current_stock_account_balance_usd                1618.0
    cash_card_enabled                                    No
    direct_deposit_enabled                               No
    cash_boost_used                                      No
    most_interacted_user_index                          529
    user_occupation                            Entrepreneur
    location                                     Washington
    most_used_cash_app_feature               Direct Deposit
    account_age_yr                                        3
    most_interacted_user_id                      aL8IUZbBDi
    user_occupation_encoded                               5
    most_used_cash_app_feature_encoded                    2
    gender_encoded                                        0
    cash_card_enabled_encoded                             0
    direct_deposit_enabled_encoded                        0
    cash_boost_used_encoded                               0
    Name: 1772, dtype: object
```

```
# Check recommender user for Manhanttan distance
```

```
recommender_user_id = ordered_customers_man[1][0]
recommended_user = cash_friends.iloc[recommender_user_id]
recommended_user
```

```
    user_id                                      FeKVVsuTml
    account_creation_date               2020-06-16 00:00:00
    gender                                           Female
    count_num_transactions_last_yr                       16
    sum_amount_spent_all_time_usd                    1377.0
    current_cash_account_balance_usd                  698.0
    current_bitcoin_account_balance_btc                2.04
    current_stock_account_balance_usd                1618.0
    cash_card_enabled                                    No
    direct_deposit_enabled                               No
    cash_boost_used                                      No
    most_interacted_user_index                          529
    user_occupation                            Entrepreneur
    location                                     Washington
    most_used_cash_app_feature               Direct Deposit
    account_age_yr                                        3
    most_interacted_user_id                      aL8IUZbBDi
    user_occupation_encoded                               5
    most_used_cash_app_feature_encoded                    2
    gender_encoded                                        0
    cash_card_enabled_encoded                             0
    direct_deposit_enabled_encoded                        0
    cash_boost_used_encoded                               0
    Name: 1772, dtype: object
```