Ubank Analisis del comportamiento de ahorros de los clientes

Libraries

```
In [7]: import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt
        from plotly.subplots import make_subplots
        import plotly.express as px
        import plotly.graph_objects as go
        import scipy.stats as stats
```

Lectura de datos de MySQL

```
In [3]: from sqlalchemy import create_engine
        import pandas as pd
        db_connection = create_engine('mysql+//dbadmin:IE8kdKO89cB79bUH#mydbinstance1.cqornbgJnzt3.us-east-1.rds.amazonaws.com/mydb')
        catalog = pd.read_sql("select * from catalog", con=db_connection)
        test_projects = pd.read_sql("select * from test_projects", con=db_connection)
        test_rules = pd.read_sql("select * from test_rules", con=db_connection)
        test_transactions = pd.read_sql("select * from test_transactions", con=db_connection)
```

Ordenamiento de los datos

Se ordenaron los datos para poder realizar un analisis exploratorio de la informacion y así poder extraer algunos punto importantes sobre el comportamiento de los clientes y sus metas de ahorros

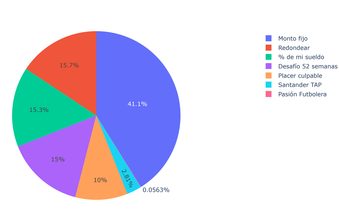
```
In [4]: test_rules_comp = pd.merge(left=test_rules, right=catalog, left_on='rule_type_id', right_on='id')
        test_projects_comp = pd.merge(left=test_projects, right=catalog, left_on='project_category_id', right_on='id')
        project_rule = pd.merge(left=test_projects_comp, right=test_rules_comp, left_on='project_id', right_on='project_id')
        total_test = pd.merge(left=test_transactions, right=project_rule, left_on='user_id', right_on='user_id')
```

```
In [5]: test_use = total_test[['user_id', 'description', 'transaction_date', 'goal_date','amount_x', 'total', 'name_y', 'categories', 'name']]
        test_use = test_use.rename(columns={'name_y': 'project_category', 'name':'rule_category', 'amount_x':'amount'})
```

Como se puede observar en la grafica de pie, el tipo de rule que más se utiliza por parte de los usuarios es "Monto Fijo" con un 41.1%; las reglas que le siguen, son "Redondeo", "% de mi sueldo", "Desafío 52 semanas", con 15.7%, 15.3% y 15% respectivamente.

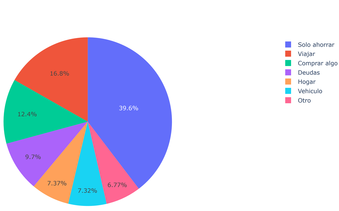
```
In [8]: pie_data_rule = test_use.groupby('rule_category').count().reset_index()
        pie_data_project = test_use.groupby('project_category').count().reset_index()

        fig = px.pie(pie_data_rule, values='amount', names='rule_category', title='Rule frecuency')
        fig.show()
```



En cuanto a la categoria de proyecto que más se utiliza es "Solo ahorrar" el tipo de proyecto o meta que más se utiliza con un 39.6% de los usuarios. Así mismo, el proyecto "viajar" es el segundo con un 16.8% de usuarios que lo plantean como una meta de ahorro.

```
In [9]: fig = px.pie(pie_data_project, values='amount', names='project_category', title='Project frecuency')
        fig.show()
```




```
In [11]: pie_data_rule_project = test_use.groupby(['project_category', 'rule_category']).count().reset_index()
         fig = px.sunburst(pie_data_rule_project, path=['project_category', 'rule_category'], values='amount', title='Rule per Project')
         fig.show()
```

En la imagen podemos observar cual es la distribucion del tipo de regla segun el tipo de proyecto utilizado; en donde se puede observar que para la categoria de proyecto "Solo ahorrar", la regla más frecuente es "Monto fijo".

Ahorro

```
In [12]: test_projects_comp = pd.merge(left=test_projects, right=catalog, left_on='project_category_id', right_on='id')
         total_test = pd.merge(left=test_transactions, right=test_projects_comp, left_on='user_id', right_on='user_id')
         total_test = total_test[['user_id','description', 'transaction_date', 'amount', 'project_id', 'goal_date', 'total', 'name_y']]
         total_test = total_test.rename(columns={'name_y': 'project_category'})
         total_test['transaction_date'] = pd.to_datetime(total_test['transaction_date'], format='%Y/%m/%d')
         total_test['goal_date'] = pd.to_datetime(total_test['goal_date'], format='%Y/%m/%d')
         total_test = total_test.sort_values('transaction_date')
         total_test = total_test.reset_index()
```

```
In [13]: start_date = total_test.groupby('project_id').first().reset_index()[['project_id','transaction_date']].rename(columns={'transaction_date': 'start_date'})
         total_test = pd.merge(left=total_test, right=start_date, left_on='project_id', right_on='project_id').drop('index', axis=1)

         total_test_success = total_test.groupby(['project_id', 'project_category', 'total', 'goal_date','start_date']).sum().reset_index()
```

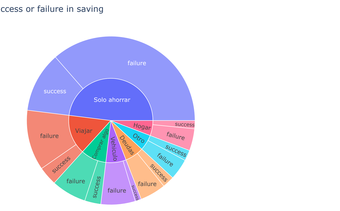
```
In [14]: rating = []
         for goal, gain  in zip(total_test_success.total, total_test_success.amount):
           if goal < gain :
             rating.append('success')
           elif goal > gain:
             rating.append('failure')
         total_test_success['success'] = rating
```

A continuación se presenta, la porcion de usuarios que lograrón o no su meta de ahorro en base al tiempo trazado, cabe resaltar que las fechas "goal_date" tienen valores a futuro, por lo tanto habrán usuarios que contaran como si no hubieran logrado su meta de trabajo, como se muestra en:

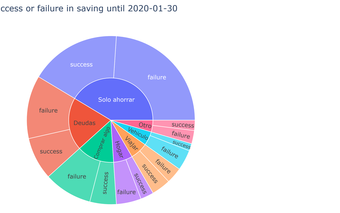
El la figura se puede observar que existe una mayor cantidad de usuarios que no lográn su meta de ahorro en comparacion de los que sí lo lograron, para cada tipo de proyecto que se establecieron los usarios.

Sin embargo, como se menciono esto se puede deber a que las metas trazadas son a largo plazo, y todavía no se ha llegado a su tiempo límite, por lo que sera necesario aplicar un filtro para poder observar aquellas metas de corto plazo.

```
In [15]: pie_data_success = total_test_success.groupby(['project_category', 'success']).count().reset_index()
         fig = px.sunburst(pie_data_success, path=['project_category', 'success'], values='amount', title='Project frequency with success or failure in saving')
         fig.show()
```




```
In [16]: success_actual = total_test_success[total_test_success['goal_date'] <= test_transactions['transaction_date'].max()].reset_index()
         pie_data_success_actual = success_actual.groupby(['project_category', 'success']).count().reset_index()
         fig = px.sunburst(pie_data_success_actual, path=['project_category', 'success'], values='amount', title='Project frequency with success or failure in saving until 2020-01-30')
         fig.show()
```



Sin embargo, al trazar como fecha maxima el 30 de enero del 2020 (ultima transaccion registrada en la base de datos), se puede hacer un filtro para establecer aquellos "goal_date" que no superen esa data; no obstante se puede observar que con dicho filtro, aun así existe más usuarios que no logran su meta de ahorra que lo que lo lograron, como se observa en Solo ahorrar; sin embargo, para el caso del tipo de proyecto "viajar" se puede observar que los usuarios lograron superar su meta de ahorro, lo que puede significar que un proyecto de tipo viaje resulta más facil de alcanzar a corto tiempo a diferencia del resto de proyectos como "hogar", "comprar algo".

Conclusiones

- Para realizar un analisis del comportamiento de ahorro de los usuarios de la aplicacion se utilizaron como datos las fechas, los tipos de proyecto y los tipos de reglas.
- Se tuvo que cambiar la estrategia para analizar dicha información para poder obtener información más concreta sobre el comportamiento que tienen los usuarios en cuanto su metodos de ahorro
- Con este análisis se puede observar que Viajar es una de las metas que tiene mayor probabilidad de lograrse en comparacion de otros tipos de proyecto, esto puede deberse a que son metas de corto plazo.