Banking Insights for Mining Communities – Jwaneng Branch Case Study#

Introduction

This project analyzes synthetic banking transactions to generate actionable insights for a mining-town branch of FNB Botswana, specifically Jwaneng. Mining communities like Jwaneng have unique payroll cycles, SME activity linked to mine contracts, and digital banking adoption patterns. Understanding these trends can help optimize branch operations, improve SME credit offerings, and enhance customer digital engagement.

DataSet Description

The base dataset is BankSim, a synthetic bank payments simulator developed for fraud detection research. BankSim contains 594,643 transaction records over 180 days, including normal payments and injected fraudulent transactions. It is fully anonymized and publicly available, making it ideal for experimentation and analysis without violating privacy.

BankSim was chosen because it provides realistic banking transactions with sufficient volume and variability, enabling the simulation of mining-town financial behaviors such as payroll-driven transaction spikes, SME loan activity, and digital banking channel usage.

Data Manipulation

```
# ------------------------------
# BANKSIM – JWANENG CASE STUDY
# ------------------------------

# 1️⃣ Import Libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import random
import os
import calendar
from datetime import datetime, timedelta

# Optional for nicer plots
sns.set(style="whitegrid")
plt.rcParams["figure.figsize"] = (12,6)

# ------------------------------
# 2️⃣ Load Dataset
# ------------------------------
df = pd.read_csv("/kaggle/input/banksim1/bs140513_032310.csv")  # adjust path if needed
df.info()
df.head()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 594643 entries, 0 to 594642
Data columns (total 10 columns):
 #   Column       Non-Null Count   Dtype
---  ------       --------------   -----
 0   step         594643 non-null  int64
 1   customer     594643 non-null  object
 2   age          594643 non-null  object
 3   gender       594643 non-null  object
 4   zipcodeOri   594643 non-null  object
 5   merchant     594643 non-null  object
 6   zipMerchant  594643 non-null  object
 7   category     594643 non-null  object
 8   amount       594643 non-null  float64
 9   fraud        594643 non-null  int64
dtypes: float64(1), int64(2), object(7)
memory usage: 45.4+ MB
```

	step	customer	age	gender	zipcodeOri	merchant	zipMerchant	category	amount	fraud
0	0	'C1093826151'	'4'	'M'	'28007'	'M348934600'	'28007'	'es_transportation'	4.55	0
1	0	'C352968107'	'2'	'M'	'28007'	'M348934600'	'28007'	'es_transportation'	39.68	0
2	0	'C2054744914'	'4'	'F'	'28007'	'M1823072687'	'28007'	'es_transportation'	26.89	0
3	0	'C1760612790'	'3'	'M'	'28007'	'M348934600'	'28007'	'es_transportation'	17.25	0
4	0	'C757503768'	'5'	'M'	'28007'	'M348934600'	'28007'	'es_transportation'	35.72	0

```
# ----------------------------
# 3️⃣ Basic Cleaning
# ----------------------------
# Strip quotes from customer and age columns
df['customer'] = df['customer'].str.replace("'", "")
df['age'] = df['age'].str.replace("'", "")

# Convert step to datetime (assume step = days from Jan 1)
start_date = datetime(2025, 1, 1)
df['date'] = df['step'].apply(lambda x: start_date + timedelta(days=x))

# Check for duplicates
df = df.drop_duplicates()

# Reset index
df.reset_index(drop=True, inplace=True)
```

```
# ----------------------------
# 4️⃣ Add Jwaneng-specific fields
# ----------------------------

# 4a) Mark some customers as mine employees
mine_employee_pct = 0.3
df['is_mine_employee'] = df['customer'].apply(lambda x: np.random.rand() < mine_employee_pct)

# 4b) Add synthetic payroll transactions
payroll_transactions = []
salary_min, salary_max = 8000, 28000

for emp in df[df['is_mine_employee']]['customer'].unique():
    for month in range(1, 13):  # months 1-12
        last_day = calendar.monthrange(2025, month)[1]  # last valid day
        for day in [15, 30]:  # payroll dates
            payroll_day = min(day, last_day)
            payroll_transactions.append({
                'step': None,
                'customer': emp,
                'age': np.random.choice(df['age']),
                'gender': np.random.choice(df['gender']),
                'zipcodeOri': 'Jwaneng',
                'merchant': 'Payroll',
                'zipMerchant': 'JWN001',
                'category': 'salary',
                'amount': round(random.uniform(salary_min, salary_max),2),
                'fraud': 0,
                'date': datetime(2025, month, payroll_day),
                'is_mine_employee': True
            })

payroll_df = pd.DataFrame(payroll_transactions)
df = pd.concat([df, payroll_df], ignore_index=True)

# 4c) Add SMEs
num_smes = 50
sme_list = [f'SME{i:03d}' for i in range(1, num_smes+1)]
sme_industries = ['logistics', 'catering', 'supplies', 'cleaning']

sme_data = []
for sme in sme_list:
    loan_amount = random.randint(20000, 500000)
    repayment_status = random.choices(['on_time','late','default'], weights=[0.7,0.2,0.1])[0]
    credit_score = random.randint(350,900)
    industry = random.choice(sme_industries)
    sme_data.append({
        'sme_id': sme,
        'industry': industry,
        'loan_amount': loan_amount,
        'repayment_status': repayment_status,
        'credit_score': credit_score
    })

sme_df = pd.DataFrame(sme_data)

# Assign SMEs to random customers
sme_customers = np.random.choice(df['customer'].unique(), num_smes, replace=False)
sme_df['customer'] = sme_customers

# 4d) Add Digital Banking Channels
channels = ['ATM', 'Branch', 'Mobile', 'USSD']
df['channel'] = df.apply(lambda x: np.random.choice(channels, p=[0.3,0.3,0.3,0.1]), axis=1)
```

```
# Increase ATM/Branch usage for payroll transactions
df.loc[df['category']=='salary', 'channel'] = np.random.choice(['ATM','Branch'], p=[0.5,0.5], size=len(df[df['category']==

# ----------------------------
# 5️⃣ Fix dates for all transactions
# ----------------------------
start_date = datetime(2025,1,1)

# Fill NaT in 'date' using 'step' for original transactions
df['date'] = df.apply(
    lambda row: row['date'] if pd.notnull(row['date']) else start_date + timedelta(days=int(row['step'])),
    axis=1
)

# ----------------------------
# 6️⃣ Final Checks
# ----------------------------
print(df.info())
print(df.head())
print(sme_df.head())

# ----------------------------
# 7️⃣ Save Prepared Datasets
# ----------------------------
df.to_csv("synthetic_jwaneng_prepared.csv", index=False)
sme_df.to_csv("sme_jwaneng.csv", index=False)
print(f"Datasets saved in: {os.getcwd()}")
```

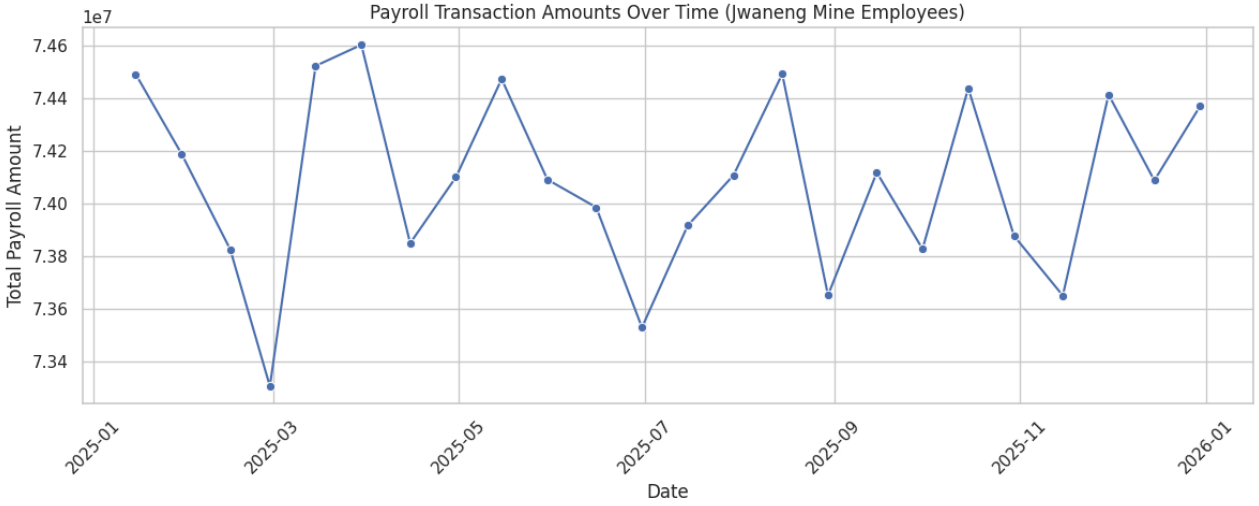
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 693283 entries, 0 to 693282
Data columns (total 13 columns):
 #   Column            Non-Null Count   Dtype
---  ------            --------------   -----
 0   step              594643 non-null  object
 1   customer          693283 non-null  object
 2   age               693283 non-null  object
 3   gender            693283 non-null  object
 4   zipcodeOri        693283 non-null  object
 5   merchant          693283 non-null  object
 6   zipMerchant       693283 non-null  object
 7   category          693283 non-null  object
 8   amount            693283 non-null  float64
 9   fraud             693283 non-null  int64
 10  date              693283 non-null  datetime64[ns]
 11  is_mine_employee  693283 non-null  bool
 12  channel           693283 non-null  object
dtypes: bool(1), datetime64[ns](1), float64(1), int64(1), object(9)
memory usage: 64.1+ MB
None
  step     customer age gender zipcodeOri       merchant zipMerchant  \
0    0  C1093826151   4    'M'    '28007'   'M348934600'     '28007'
1    0   C352968107   2    'M'    '28007'   'M348934600'     '28007'
2    0  C2054744914   4    'F'    '28007'  'M1823072687'     '28007'
3    0  C1760612790   3    'M'    '28007'   'M348934600'     '28007'
4    0   C757503768   5    'M'    '28007'   'M348934600'     '28007'

              category  amount  fraud       date  is_mine_employee channel
0  'es_transportation'    4.55      0 2025-01-01             False    USSD
1  'es_transportation'   39.68      0 2025-01-01             False     ATM
2  'es_transportation'   26.89      0 2025-01-01             False  Mobile
3  'es_transportation'   17.25      0 2025-01-01             False     ATM
4  'es_transportation'   35.72      0 2025-01-01             False     ATM
   sme_id  industry  loan_amount repayment_status  credit_score     customer
0  SME001  cleaning       460128             late           565  C1351458855
1  SME002  catering        63035          on_time           860  C2001133685
2  SME003  cleaning       402467             late           590  C1058213633
3  SME004  catering       273548          on_time           447   C469822988
4  SME005  catering       424555          on_time           517   C746862122
Datasets saved in: /kaggle/working
```

Data Analysis

```
# ----------------------------
# 2️⃣ Load Prepared Datasets
# ----------------------------
df = pd.read_csv("synthetic_jwaneng_prepared.csv", parse_dates=['date'])
sme_df = pd.read_csv("sme_jwaneng.csv")

# ----------------------------
# 3️⃣ Payroll Transaction Analysis
# ----------------------------
# Filter payroll transactions
payroll_df = df[df['category'] == 'salary']
```

```
# Aggregate by date
payroll_daily = payroll_df.groupby('date')['amount'].sum().reset_index()

# Plot payroll spikes
plt.figure(figsize=(12,5))
sns.lineplot(data=payroll_daily, x='date', y='amount', marker='o')
plt.title("Payroll Transaction Amounts Over Time (Jwaneng Mine Employees)")
plt.xlabel("Date")
plt.ylabel("Total Payroll Amount")
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()

# Optional: Top 10 highest payroll days
top_paydays = payroll_daily.sort_values(by='amount', ascending=False).head(10)
print("Top 10 payroll days by total amount:")
print(top_paydays)
```

```
/usr/local/lib/python3.11/dist-packages/seaborn/_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated and will
  with pd.option_context('mode.use_inf_as_na', True):
/usr/local/lib/python3.11/dist-packages/seaborn/_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated and will
  with pd.option_context('mode.use_inf_as_na', True):
```




```
Top 10 payroll days by total amount:
         date       amount
5  2025-03-30  74601561.50
4  2025-03-15  74521511.52
14 2025-08-15  74490490.18
0  2025-01-15  74490175.78
8  2025-05-15  74471561.00
18 2025-10-15  74436541.37
21 2025-11-30  74412341.94
23 2025-12-30  74368403.58
1  2025-01-30  74188476.89
16 2025-09-15  74116866.94
```

Payroll Transaction Spikes

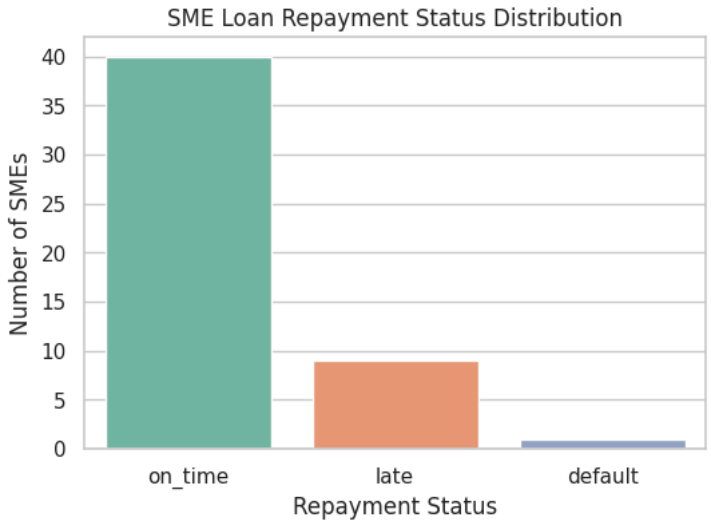
The line chart above shows the total payroll transaction amounts for mine employees over time. Notice the clear spikes on the 15th and end of each month, corresponding to payday cycles.

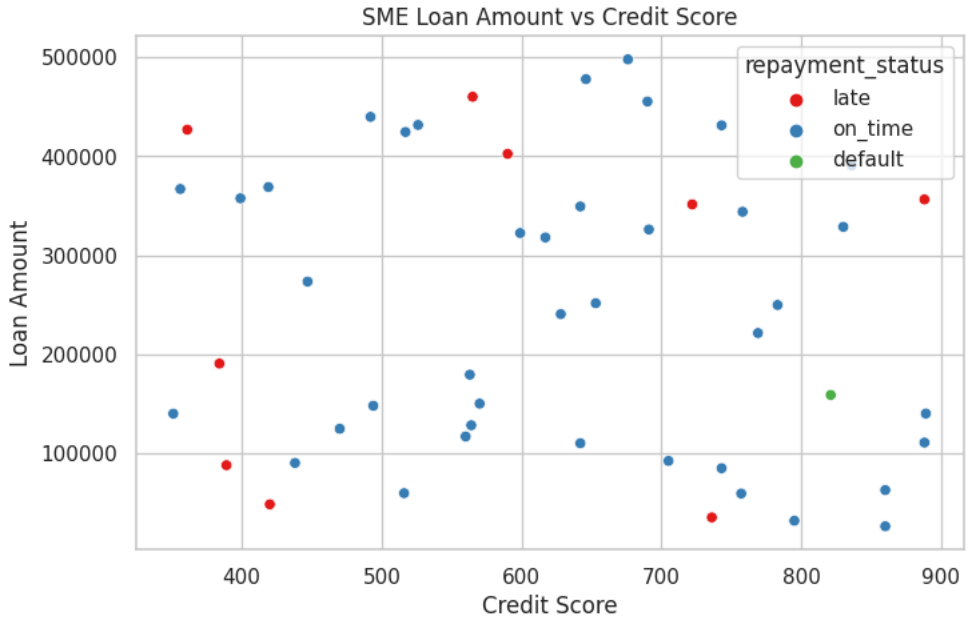
Insight:

- FNB Jwaneng can anticipate high transaction volumes around these dates.
- Scheduling staff and cash/ATM replenishment around these dates can improve service efficiency.
- The bank could offer payday-linked micro-loans or digital savings campaigns to employees.

```
# ----------------------------
# 4️⃣ SME Financing Trends
# ----------------------------
# Count SMEs by repayment status
repayment_counts = sme_df['repayment_status'].value_counts()

plt.figure(figsize=(6,4))
```

```
sns.barplot(x=repayment_counts.index, y=repayment_counts.values, palette="Set2")
plt.title("SME Loan Repayment Status Distribution")
plt.xlabel("Repayment Status")
plt.ylabel("Number of SMEs")
plt.show()

# Loan amount vs credit score scatter
plt.figure(figsize=(8,5))
sns.scatterplot(data=sme_df, x='credit_score', y='loan_amount', hue='repayment_status', palette="Set1")
plt.title("SME Loan Amount vs Credit Score")
plt.xlabel("Credit Score")
plt.ylabel("Loan Amount")
plt.show()
```





SME Loan Repayment Behavior

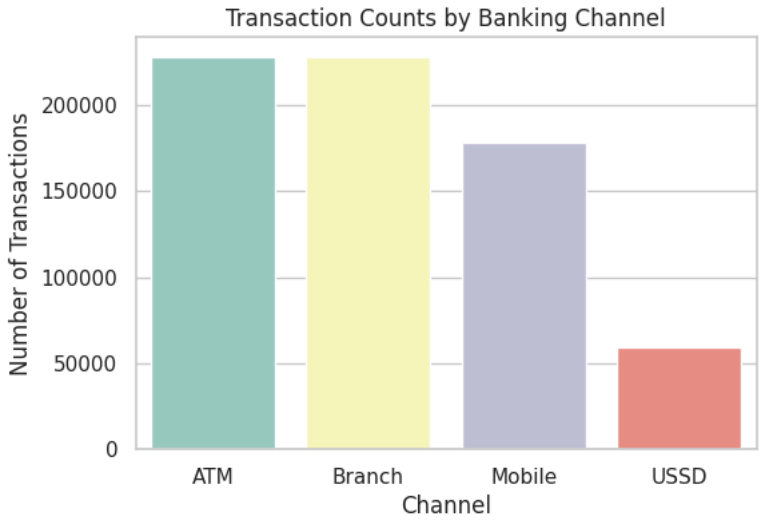
The bar chart shows the distribution of SME loan repayment status (on-time, late, default). The scatter plot shows how loan amount relates to credit score.

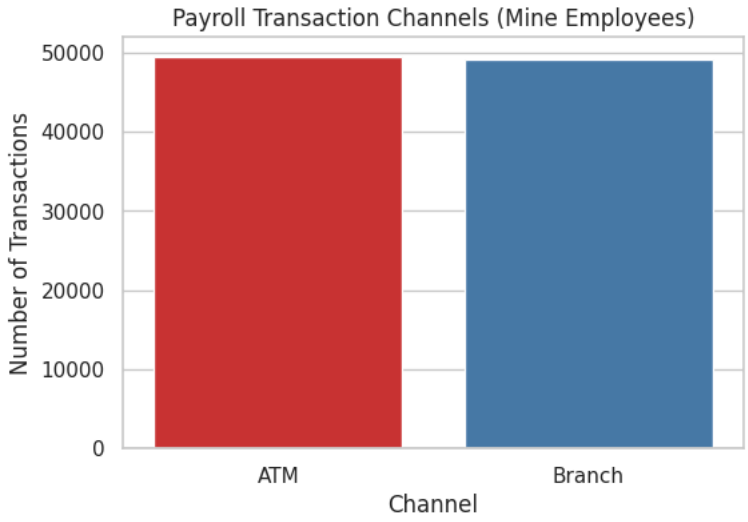
Insights:

- Most SMEs repay on time, but a small percentage are late or default.
- Higher credit scores generally correlate with timely repayment, but there are exceptions.
- FNB could use these insights to refine SME loan risk models and offer credit products tailored to industry type or repayment reliability.

```
# ----------------------------
# 5️⃣ Digital Banking Adoption
# ----------------------------
# Count transactions by channel
channel_counts = df['channel'].value_counts()
```

```
plt.figure(figsize=(6,4))
sns.barplot(x=channel_counts.index, y=channel_counts.values, palette="Set3")
plt.title("Transaction Counts by Banking Channel")
plt.xlabel("Channel")
plt.ylabel("Number of Transactions")
plt.show()

# Channel usage for payroll transactions specifically
payroll_channels = payroll_df['channel'].value_counts()

plt.figure(figsize=(6,4))
sns.barplot(x=payroll_channels.index, y=payroll_channels.values, palette="Set1")
plt.title("Payroll Transaction Channels (Mine Employees)")
plt.xlabel("Channel")
plt.ylabel("Number of Transactions")
plt.show()
```





Digital Banking Channels

The bar chart shows the number of transactions by channel, while the payroll-specific chart highlights how mine employees use ATMs and branches for salary withdrawals.

Insights:

- ATM and branch usage dominates payroll withdrawals.
- Mobile and USSD channels are underutilized for salary transactions, suggesting an opportunity to promote digital banking adoption among mine employees.
- FNB could implement targeted campaigns or incentives to increase mobile banking usage, reducing branch congestion and operational costs.

```
# ----------------------------
# 6️⃣ Summary Insights (print)
# ----------------------------
```

```
num_mine_employees = df['is_mine_employee'].sum()
num_smes = len(sme_df)
total_transactions = len(df)
total_payroll = payroll_df['amount'].sum()

print(f"Total transactions: {total_transactions}")
print(f"Mine employees: {num_mine_employees}")
print(f"Total payroll distributed: BWP {total_payroll:,.2f}")
print(f"Total SMEs analyzed: {num_smes}")
```

```
Total transactions: 693283
Mine employees: 277554
Total payroll distributed: BWP 1,777,892,208.79
Total SMEs analyzed: 50
```