# Mpesa statement - from pdf to csv file

November 5, 2025

[1]:
```
import pdfplumber #library for reading pdf files
import pandas as pd

# === USER INPUT ===
pdf_path = "path of where the pdf file is stored in your pc"  # Replace with␣
↪your actual file name or path
pdf_password = "shared by safaricom as sms"  # Replace with the correct password

# === STEP 1: Open the PDF ===
all_rows = []

with pdfplumber.open(pdf_path, password=pdf_password) as pdf:
    for page in pdf.pages:
        tables = page.extract_tables()
        for table in tables:
            for row in table:
                # Skip header rows or empty rows
                if row[0] == "Receipt No" or row[0] is None:
                    continue
                all_rows.append(row)

# === STEP 2: Create DataFrame ===
columns = ["Receipt No", "Completion time", "Details", "Transaction status",␣
↪"paid in", "Withdrawn", "Balance"]
df = pd.DataFrame(all_rows, columns=columns)

# === STEP 3: Save to CSV ===
#df.to_csv("mpesa_transactions.csv", index=False)

#print("[U+2705] M-Pesa transactions saved to 'mpesa_transactions.csv'")
```

[9]:
```
#The fisrt 9 rows are the summary(e.g total received, total send etc) of the␣
↪whole mpesa statement and hence need of dropping
df = df.drop(df.head(9).index)
```

[10]:
```
#The names of the columns of the statement were repeating themselves after every␣
↪page. I therefore renamed the first column to allow me to drop the rest
```

```
# Print current column names to verify
print("Original columns:", df.columns.tolist())

# Rename the first column explicitly
df.columns = ["Receipt_No"] + df.columns.tolist()[1:]
```

```
Original columns: ['Receipt_No', 'Completion time', 'Details', 'Transaction
status', 'paid in', 'Withdrawn', 'Balance']
```

[23]:
```
# === Drop rows where the first column contains "Receipt No." ===
first_column = df.columns[0]  # Automatically get the name of the first column
df = df[~df[first_column].astype(str).str.contains("Receipt No.", case=False,␣
↪na=False)]
```

[24]:
```
# ===  Drop rows where the second column contains "None" ===
first_column = df.columns[1]  # Automatically get the name of the first column
df = df[~df[first_column].astype(str).str.contains("None", case=False, na=False)]
```

[25]:
```
# ===  Drop rows where the first column contains "Statement Verification Code"␣
↪===
first_column = df.columns[0]  # Automatically get the name of the first column
df = df[~df[first_column].astype(str).str.contains("Statement Verification␣
↪Code", case=False, na=False)]
```

[26]:
```
# ===  Drop rows where the first column contains "UN7KKADG" ===
first_column = df.columns[0]  # Automatically get the name of the first column
df = df[~df[first_column].astype(str).str.contains("UN7KKADG", case=False,␣
↪na=False)]
```

[27]:
```
# ===  Drop rows where the first column contains "NaN" ===
first_column = df.columns[0]  # Automatically get the name of the first column
df = df[~df[first_column].astype(str).str.contains("NaN", case=False, na=False)]
```

[28]:
```
# ===  Save to CSV ===
df.to_csv("M-pesa_transactions.csv", index=False)
```