```
from google.colab import files
uploaded = files.upload()

```

```
Choose Files Fraud Dete… Dataset.csv
• Fraud Detection Dataset.csv(text/csv) - 3692523 bytes, last modified: 4/2/2025 - 100% done
Saving Fraud Detection Dataset.csv to Fraud Detection Dataset.csv
```

```
import pandas as pd

df = pd.read_csv('Fraud Detection Dataset.csv')
```

```
print("🔹 First 5 rows of the dataset:")
print(df.head())
print("\n🔹 Columns:")
print(df.columns.tolist())

```

```
🔹 First 5 rows of the dataset:
  Transaction_ID  User_ID  Transaction_Amount Transaction_Type  \
0             T1     4174             1292.76   ATM Withdrawal
1             T2     4507             1554.58   ATM Withdrawal
2             T3     1860             2395.02   ATM Withdrawal
3             T4     2294              100.10     Bill Payment
4             T5     2130             1490.50      POS Payment

   Time_of_Transaction Device_Used       Location  \
0                 16.0      Tablet  San Francisco
1                 13.0      Mobile       New York
2                  NaN      Mobile            NaN
3                 15.0     Desktop        Chicago
4                 19.0      Mobile  San Francisco

   Previous_Fraudulent_Transactions  Account_Age  \
0                                 0          119
1                                 4           79
2                                 3          115
3                                 4            3
4                                 2           57

   Number_of_Transactions_Last_24H Payment_Method  Fraudulent
0                               13     Debit Card           0
1                                3    Credit Card           0
2                                9            NaN           0
3                                4            UPI           0
4                                7    Credit Card           0

🔹 Columns:
['Transaction_ID', 'User_ID', 'Transaction_Amount', 'Transaction_Type', 'Time_of_Transaction', 'Device_Used', 'Location', 'Previous_
```

```
# Create subsets
df_part1 = df.iloc[:5].copy()
df_part2 = df.iloc[5:10].copy()
```

```
# Concat vertically
concat_vertical = pd.concat([df_part1, df_part2], axis=0)
print("\n Concat Vertical:")
print(concat_vertical)

```

```
 Concat Vertical:
  Transaction_ID  User_ID  Transaction_Amount Transaction_Type  \
0             T1     4174             1292.76   ATM Withdrawal
1             T2     4507             1554.58   ATM Withdrawal
2             T3     1860             2395.02   ATM Withdrawal
3             T4     2294              100.10     Bill Payment
4             T5     2130             1490.50      POS Payment
5             T6     2095             2372.04   ATM Withdrawal
6             T7     4772              544.81     Bill Payment
7             T8     4092              635.75   ATM Withdrawal
8             T9     2638             2318.87    Bank Transfer
9            T10     3169             3656.17     Bill Payment

   Time_of_Transaction Device_Used       Location  \
0                 16.0      Tablet  San Francisco
1                 13.0      Mobile       New York
2                  NaN      Mobile            NaN
3                 15.0     Desktop        Chicago
4                 19.0      Mobile  San Francisco
5                 15.0     Desktop         Boston
6                  2.0      Tablet         Boston
```

```
7                13.0      Tablet         Boston
8                 NaN      Mobile  San Francisco
9                 3.0      Mobile        Chicago

   Previous_Fraudulent_Transactions  Account_Age  \
0                                 0          119
1                                 4           79
2                                 3          115
3                                 4            3
4                                 2           57
5                                 3           96
6                                 3            6
7                                 2           13
8                                 4          110
9                                 4           66

   Number_of_Transactions_Last_24H Payment_Method  Fraudulent
0                               13     Debit Card           0
1                                3    Credit Card           0
2                                9            NaN           0
3                                4            UPI           0
4                                7    Credit Card           0
5                               14    Credit Card           0
6                                9            UPI           1
7                               10     Debit Card           0
8                               12     Debit Card           0
9                                3    Net Banking           0
```

```
# Concat horizontally
concat_horizontal = pd.concat([df_part1.reset_index(drop=True), df_part2.reset_index(drop=True)], axis=1)
print("\n Concat Horizontal:")
print(concat_horizontal)
```

```
 Concat Horizontal:
  Transaction_ID  User_ID Transaction_Amount  Transaction_Type  \
0             T1     4174            1292.76    ATM Withdrawal
1             T2     4507            1554.58    ATM Withdrawal
2             T3     1860            2395.02    ATM Withdrawal
3             T4     2294             100.10      Bill Payment
4             T5     2130            1490.50       POS Payment

  Time_of_Transaction  Device_Used       Location  \
0                16.0       Tablet  San Francisco
1                13.0       Mobile       New York
2                 NaN       Mobile            NaN
3                15.0      Desktop        Chicago
4                19.0       Mobile  San Francisco

  Previous_Fraudulent_Transactions  Account_Age  \
0                                0          119
1                                4           79
2                                3          115
3                                4            3
4                                2           57

  Number_of_Transactions_Last_24H  ... Transaction_Amount  Transaction_Type  \
0                              13  ...            2372.04    ATM Withdrawal
1                               3  ...             544.81      Bill Payment
2                               9  ...             635.75    ATM Withdrawal
3                               4  ...            2318.87     Bank Transfer
4                               7  ...            3656.17      Bill Payment

  Time_of_Transaction  Device_Used       Location  \
0                15.0      Desktop         Boston
1                 2.0       Tablet         Boston
2                13.0       Tablet         Boston
3                 NaN       Mobile  San Francisco
4                 3.0       Mobile        Chicago

  Previous_Fraudulent_Transactions  Account_Age  \
0                                3           96
1                                3            6
2                                2           13
3                                4          110
4                                4           66

  Number_of_Transactions_Last_24H Payment_Method  Fraudulent
0                              14    Credit Card           0
1                               9            UPI           1
2                              10     Debit Card           0
3                              12     Debit Card           0
4                               3    Net Banking           0

[5 rows x 24 columns]
```

```
# Append (equivalent to concat)
appended_df = pd.concat([df_part1, df_part2])
```

```
print("\n Appended DataFrame:")
print(appended_df)
```

```
 Appended DataFrame:
  Transaction_ID  User_ID  Transaction_Amount Transaction_Type  \
0             T1     4174             1292.76   ATM Withdrawal
1             T2     4507             1554.58   ATM Withdrawal
2             T3     1860             2395.02   ATM Withdrawal
3             T4     2294              100.10     Bill Payment
4             T5     2130             1490.50      POS Payment
5             T6     2095             2372.04   ATM Withdrawal
6             T7     4772              544.81     Bill Payment
7             T8     4092              635.75   ATM Withdrawal
8             T9     2638             2318.87    Bank Transfer
9            T10     3169             3656.17     Bill Payment

   Time_of_Transaction Device_Used       Location  \
0                 16.0      Tablet  San Francisco
1                 13.0      Mobile       New York
2                  NaN      Mobile            NaN
3                 15.0     Desktop        Chicago
4                 19.0      Mobile  San Francisco
5                 15.0     Desktop         Boston
6                  2.0      Tablet         Boston
7                 13.0      Tablet         Boston
8                  NaN      Mobile  San Francisco
9                  3.0      Mobile        Chicago

   Previous_Fraudulent_Transactions  Account_Age  \
0                                 0          119
1                                 4           79
2                                 3          115
3                                 4            3
4                                 2           57
5                                 3           96
6                                 3            6
7                                 2           13
8                                 4          110
9                                 4           66

   Number_of_Transactions_Last_24H Payment_Method  Fraudulent
0                               13     Debit Card           0
1                                3    Credit Card           0
2                                9            NaN           0
3                                4            UPI           0
4                                7    Credit Card           0
5                               14    Credit Card           0
6                                9            UPI           1
7                               10     Debit Card           0
8                               12     Debit Card           0
9                                3    Net Banking           0
```

```
# Merge based on Transaction_ID
df_merge1 = df[['Transaction_ID', 'User_ID', 'Transaction_Amount']].iloc[:5].copy()
df_merge2 = df[['Transaction_ID', 'Transaction_Type', 'Fraudulent']].iloc[:5].copy()
merged_df = pd.merge(df_merge1, df_merge2, on='Transaction_ID', how='inner')
print("\n Merged DataFrame:")
print(merged_df)
```

```
 Merged DataFrame:
  Transaction_ID  User_ID  Transaction_Amount Transaction_Type  Fraudulent
0             T1     4174             1292.76   ATM Withdrawal           0
1             T2     4507             1554.58   ATM Withdrawal           0
2             T3     1860             2395.02   ATM Withdrawal           0
3             T4     2294              100.10     Bill Payment           0
4             T5     2130             1490.50      POS Payment           0
```

```
# Join using index
df_merge1_idx = df_merge1.set_index('Transaction_ID')
df_merge2_idx = df_merge2.set_index('Transaction_ID')
joined_df = df_merge1_idx.join(df_merge2_idx, how='inner')
print("\n Joined DataFrame (on index):")
print(joined_df)
```

```
 Joined DataFrame (on index):
                User_ID  Transaction_Amount Transaction_Type  Fraudulent
Transaction_ID
T1                 4174             1292.76   ATM Withdrawal           0
T2                 4507             1554.58   ATM Withdrawal           0
T3                 1860             2395.02   ATM Withdrawal           0
T4                 2294              100.10     Bill Payment           0
T5                 2130             1490.50      POS Payment           0
```

```
#  STEP 3: AGGREGATION AND GROUPING

agg_result = df.groupby('Transaction_Type')['Transaction_Amount'].agg(['mean', 'max', 'min'])
print("\n Aggregation (mean, max, min) by Transaction Type:")
print(agg_result)

```

```
 Aggregation (mean, max, min) by Transaction Type:
                         mean      max   min
Transaction_Type
ATM Withdrawal    2977.734152  49997.8  5.08
Bank Transfer     3032.651571  49997.8  5.52
Bill Payment      3038.556278  49997.8  5.23
Online Purchase   2988.188041  49997.8  5.03
POS Payment       2942.371743  49997.8  5.04
```

```
#  STEP 4: PIVOT TABLE

pivot_table = pd.pivot_table(df,
                             values='Transaction_Amount',
                             index='Device_Used',
                             columns='Payment_Method',
                             aggfunc='mean')
print("\n Pivot Table (Avg Transaction Amount by Device & Payment Method):")
print(pivot_table)
```

```
 Pivot Table (Avg Transaction Amount by Device & Payment Method):
Payment_Method  Credit Card   Debit Card  Invalid Method  Net Banking  \
Device_Used
Desktop         3084.315162  3052.959960     2993.530638  2896.305835
Mobile          3000.030658  2938.878493     3362.946958  2870.089618
Tablet          2990.772888  3007.787078     2833.454764  3018.821398
Unknown Device  3061.655559  3102.543234     2441.068000  3468.624880

Payment_Method          UPI
Device_Used
Desktop         2977.207359
Mobile          3200.024867
Tablet          2815.039924
Unknown Device  2550.925890
```

```
#  STEP 5: VECTORIZED STRING OPERATIONS

df['Location'] = df['Location'].astype(str)  # Ensure string type
contains_san = df[df['Location'].str.contains("San", case=False, na=False)]
print("\n Locations containing 'San':")
print(contains_san[['Transaction_ID', 'Location']])
```

```
 Locations containing 'San':
                     Transaction_ID       Location
Transaction_Date
2023-01-01 00:00:00             T1  San Francisco
2023-01-01 04:00:00             T5  San Francisco
2023-01-01 08:00:00             T9  San Francisco
2023-01-01 11:00:00            T12  San Francisco
2023-01-01 14:00:00            T15  San Francisco
...                            ...            ...
2028-10-24 21:00:00          T4581  San Francisco
2028-10-25 08:00:00         T18786  San Francisco
2028-10-25 12:00:00         T11626  San Francisco
2028-10-25 14:00:00         T12629  San Francisco
2028-10-25 21:00:00         T12293  San Francisco

[5985 rows x 2 columns]
```

```
#  STEP 6: WORKING WITH TIME SERIES

# Add a fake date column (since original dataset doesn't have it)
df['Transaction_Date'] = pd.date_range(start="2023-01-01", periods=len(df), freq='h')
df.set_index('Transaction_Date', inplace=True)
```

```
# Resample to get daily transaction count
daily_txn_count = df['Transaction_ID'].resample('D').count()
print("\n Daily Transaction Counts:")
print(daily_txn_count)
```

```
 Daily Transaction Counts:
Transaction_Date
2023-01-01    24
2023-01-02    24
```

```
2023-01-03    24
2023-01-04    24
2023-01-05    24
              ..
2028-10-21    24
2028-10-22    24
2028-10-23    24
2028-10-24    24
2028-10-25    24
Freq: D, Name: Transaction_ID, Length: 2125, dtype: int64
```

```
#  STEP 7: HIGH PERFORMANCE FUNCTIONS — query() & eval()

# Filter high-value transactions using query()
high_value_txns = df.query("Transaction_Amount > 2000")
print("\n High Value Transactions (Amount > 2000):")
print(high_value_txns[['Transaction_ID', 'Transaction_Amount']])
```

```
 High Value Transactions (Amount > 2000):
                     Transaction_ID  Transaction_Amount
Transaction_Date
2023-01-01 02:00:00             T3             2395.02
2023-01-01 05:00:00             T6             2372.04
2023-01-01 08:00:00             T9             2318.87
2023-01-01 09:00:00            T10             3656.17
2023-01-01 11:00:00            T12             2733.84
...                            ...                 ...
2028-10-25 17:00:00          T5873             3613.59
2028-10-25 19:00:00         T33982             3112.51
2028-10-25 20:00:00         T31261             2897.15
2028-10-25 21:00:00         T12293             2204.43
2028-10-25 22:00:00         T42287             4787.17

[29275 rows x 2 columns]
```

```
# Use eval() to compute a new 'Risk_Score'
df.eval("Risk_Score = Previous_Fraudulent_Transactions * 2 + Number_of_Transactions_Last_24H", inplace=True)
print("\nDataFrame with calculated Risk_Score:")
print(df[['Transaction_ID', 'Previous_Fraudulent_Transactions', 'Number_of_Transactions_Last_24H', 'Risk_Score']].head())

```

```
DataFrame with calculated Risk_Score:
                     Transaction_ID  Previous_Fraudulent_Transactions  \
Transaction_Date
2023-01-01 00:00:00             T1                                 0
2023-01-01 01:00:00             T2                                 4
2023-01-01 02:00:00             T3                                 3
2023-01-01 03:00:00             T4                                 4
2023-01-01 04:00:00             T5                                 2

                     Number_of_Transactions_Last_24H  Risk_Score
Transaction_Date
2023-01-01 00:00:00                               13          13
2023-01-01 01:00:00                                3          11
2023-01-01 02:00:00                                9          15
2023-01-01 03:00:00                                4          12
2023-01-01 04:00:00                                7          11
```

```
#  1. Using apply() for custom row-wise operations
def classify_transaction(row):
    if row['Transaction_Amount'] > 3000:
        return "High"
    elif row['Transaction_Amount'] > 1000:
        return "Medium"
    else:
        return "Low"

df['Transaction_Level'] = df.apply(classify_transaction, axis=1)
print("\n Transaction_Level column added using apply():")
print(df[['Transaction_ID', 'Transaction_Amount', 'Transaction_Level']].head())

```

```
 Transaction_Level column added using apply():
                     Transaction_ID  Transaction_Amount Transaction_Level
Transaction_Date
2023-01-01 00:00:00             T1             1292.76            Medium
2023-01-01 01:00:00             T2             1554.58            Medium
2023-01-01 02:00:00             T3             2395.02            Medium
2023-01-01 03:00:00             T4              100.10               Low
2023-01-01 04:00:00             T5             1490.50            Medium
```

```
#  2. Value Counts and Normalization
```

```
# 2. Value counts and normalization
fraud_counts = df['Fraudulent'].value_counts()
fraud_normalized = df['Fraudulent'].value_counts(normalize=True)

print("\n Count of Fraudulent vs Non-Fraudulent transactions:")
print(fraud_counts)

print("\n Normalized proportion (in %):")
print(fraud_normalized * 100)
```

```
 Count of Fraudulent vs Non-Fraudulent transactions:
Fraudulent
0    48490
1     2510
Name: count, dtype: int64

 Normalized proportion (in %):
Fraudulent
0    95.078431
1     4.921569
Name: proportion, dtype: float64
```

```
#  3. Crosstab between two columns
# ------------------------------------------------
fraud_device_crosstab = pd.crosstab(df['Fraudulent'], df['Device_Used'])
print("\n Crosstab of Fraudulent vs Device Used:")
print(fraud_device_crosstab)
```

```
 Crosstab of Fraudulent vs Device Used:
Device_Used  Desktop  Mobile  Tablet  Unknown Device
Fraudulent
0              15047   14808   14859            1455
1                748     806     729              75
```

```
#  4. Handling Missing Values (if any)
# ------------------------------------------------
missing_info = df.isnull().sum()
print("\n Missing values in each column:")
print(missing_info[missing_info > 0])
```

```
 Missing values in each column:
Transaction_Amount     2520
Time_of_Transaction    2552
Device_Used            2473
Payment_Method         2469
dtype: int64
```

```
# 5. Sorting
sorted_by_amount = df.sort_values(by='Transaction_Amount', ascending=False)
print("\n Top 5 transactions by amount:")
print(sorted_by_amount[['Transaction_ID', 'Transaction_Amount']].head())

```

```
 Top 5 transactions by amount:
                    Transaction_ID  Transaction_Amount
Transaction_Date
2023-11-03 04:00:00          T7349             49997.8
2024-08-10 16:00:00         T14105             49997.8
2023-11-09 14:00:00          T7503             49997.8
2023-11-09 20:00:00          T7509             49997.8
2026-07-07 21:00:00         T30814             49997.8
```

```
# 6. Duplicates Check
duplicate_rows = df[df.duplicated()]
print("\n Duplicate rows in dataset:")
print(duplicate_rows)
```

```
                         Location  Previous_Fraudulent_Transactions  \
Transaction_Date
2028-09-14 08:00:00       Seattle                                 2
2028-09-14 09:00:00  San Francisco                                4
2028-09-14 10:00:00         Miami                                 1
2028-09-14 11:00:00    Los Angeles                                 3
2028-09-14 12:00:00    Los Angeles                                 3
...                            ...                               ...
2028-10-25 17:00:00      New York                                  3
2028-10-25 18:00:00      New York                                  1
2028-10-25 19:00:00      New York                                  0
2028-10-25 20:00:00         Miami                                 1
2028-10-25 22:00:00      New York                                  2

                     Account_Age  Number_of_Transactions_Last_24H  \
Transaction_Date
2028-09-14 08:00:00           40                                7
2028-09-14 09:00:00            7                                3
2028-09-14 10:00:00           39                                6
2028-09-14 11:00:00           67                                9
2028-09-14 12:00:00           12                               10
...                           ...                              ...
2028-10-25 17:00:00           58                                7
2028-10-25 18:00:00           94                                6
2028-10-25 19:00:00            7                                8
2028-10-25 20:00:00           75                               11
2028-10-25 22:00:00          108                               14

                    Payment_Method  Fraudulent  Risk_Score Transaction_Level
Transaction_Date
2028-09-14 08:00:00            UPI           0          11            Medium
2028-09-14 09:00:00            NaN           0          11            Medium
2028-09-14 10:00:00            UPI           1           8              High
2028-09-14 11:00:00    Credit Card           0          15               Low
2028-09-14 12:00:00            NaN           0          16            Medium
...                            ...         ...         ...               ...
2028-10-25 17:00:00    Net Banking           0          13              High
2028-10-25 18:00:00            UPI           0           8            Medium
2028-10-25 19:00:00     Debit Card           0           8              High
2028-10-25 20:00:00    Net Banking           1          13            Medium
2028-10-25 22:00:00    Net Banking           0          18              High

[881 rows x 14 columns]
```

```
# 7. Binning (Creating Ranges)
# ------------------------------------------------
df['Amount_Range'] = pd.cut(df['Transaction_Amount'],
                            bins=[0, 1000, 3000, 10000],
                            labels=['Low', 'Medium', 'High'])
print("\n Amount Range created using binning:")
print(df[['Transaction_ID', 'Transaction_Amount', 'Amount_Range']].head())
```

```
 Amount Range created using binning:
                    Transaction_ID  Transaction_Amount Amount_Range
Transaction_Date
2023-01-01 00:00:00             T1             1292.76       Medium
2023-01-01 01:00:00             T2             1554.58       Medium
2023-01-01 02:00:00             T3             2395.02       Medium
2023-01-01 03:00:00             T4              100.10          Low
2023-01-01 04:00:00             T5             1490.50       Medium
```

```
#  8. Groupby Multiple Columns
multi_group = df.groupby(['Device_Used', 'Fraudulent'])['Transaction_Amount'].agg(['count', 'mean', 'sum'])
print("\n Grouped by Device and Fraudulent status:")
print(multi_group)
```

```
 Grouped by Device and Fraudulent status:
                           count         mean          sum
Device_Used    Fraudulent
Desktop        0           14301  3000.051465  42903736.00
               1             713  2889.498513   2060212.44
Mobile         0           14091  3012.135873  42444006.59
               1             765  3235.595359   2475230.45
Tablet         0           14120  2972.799101  41975923.30
               1             695  3042.683050   2114664.72
Unknown Device 0            1397  2922.764846   4083102.49
               1              70  4403.385429    308236.98
```