The Digital India program is a flagship program of the Government of India with a vision to transform India into a digitally empowered society and knowledge economy. Digital India promotes cashless transactions and aims to convert India into a cashless society.

In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
df = pd.read_csv("NDAP_REPORT_7072.csv")
```

In [3]:
```
# Display the first few rows of the data for inspection
#Scaling Factor Crores (1,00,00,000INR)
df.head(5)
```

Out[3]:

	ROWID	Name of the ministry	Name of the Project	SourceMonth	SourceYear	Digital Transaction	BHIM transaction	Debit Card	Cou
0	1	Ministry of Electronics and Information Techno...	Digital Payments	Apr - 2020	2020	303.40	99.95	20.95	I
1	2	Ministry of Electronics and Information Techno...	Digital Payments	Jun - 2017	2017	152.90	1.02	25.98	I
2	3	Ministry of Electronics and Information Techno...	Digital Payments	Jul - 2017	2017	156.57	1.14	26.13	I
3	4	Ministry of Electronics and Information Techno...	Digital Payments	May - 2021	2021	520.38	253.95	26.36	I
4	5	Ministry of Electronics and Information Techno...	Digital Payments	Sep - 2017	2017	158.34	3.08	26.64	I

In [4]:
```
# Basic summary statistics of the transaction data
df.describe()
```

Out[4]:

	ROWID	SourceYear	Digital Transaction	BHIM transaction	Debit Card	YearCode	MonthCode
count	64.000000	64.000000	64.000000	64.000000	64.000000	64.000000	64.000000
mean	32.500000	2019.421875	433.863750	171.850625	35.068125	2019.421875	201948.625000
std	18.618987	1.591692	238.489914	173.195684	5.714014	1.591692	158.427541
min	1.000000	2017.000000	152.900000	0.700000	20.950000	2017.000000	201704.000000
25%	16.750000	2018.000000	241.570000	29.547500	31.592500	2018.000000	201807.750000
50%	32.500000	2019.000000	389.500000	118.350000	35.800000	2019.000000	201911.500000
75%	48.250000	2021.000000	546.805000	256.487500	38.585000	2021.000000	202103.250000
max	64.000000	2022.000000	1066.310000	595.530000	46.370000	2022.000000	202207.000000

In [5]:

```
# Display the data types of each column
df.dtypes
```

Out[5]:

```
ROWID                      int64
Name of the ministry      object
Name of the Project       object
SourceMonth               object
SourceYear                 int64
Digital Transaction      float64
BHIM transaction         float64
Debit Card               float64
Country                   object
YearCode                   int64
Year                      object
MonthCode                  int64
Month                     object
dtype: object
```

In [6]:

```
# Display the number of rows and columns in the dataset
rows, columns = df.shape
print(f"Number of rows: {rows}, Number of columns: {columns}")
```

```
Number of rows: 64, Number of columns: 13
```

In [7]:

```
# Check for missing values in each column
missing_values = df.isnull().sum()

# Display the count of missing values in each column
missing_values
```

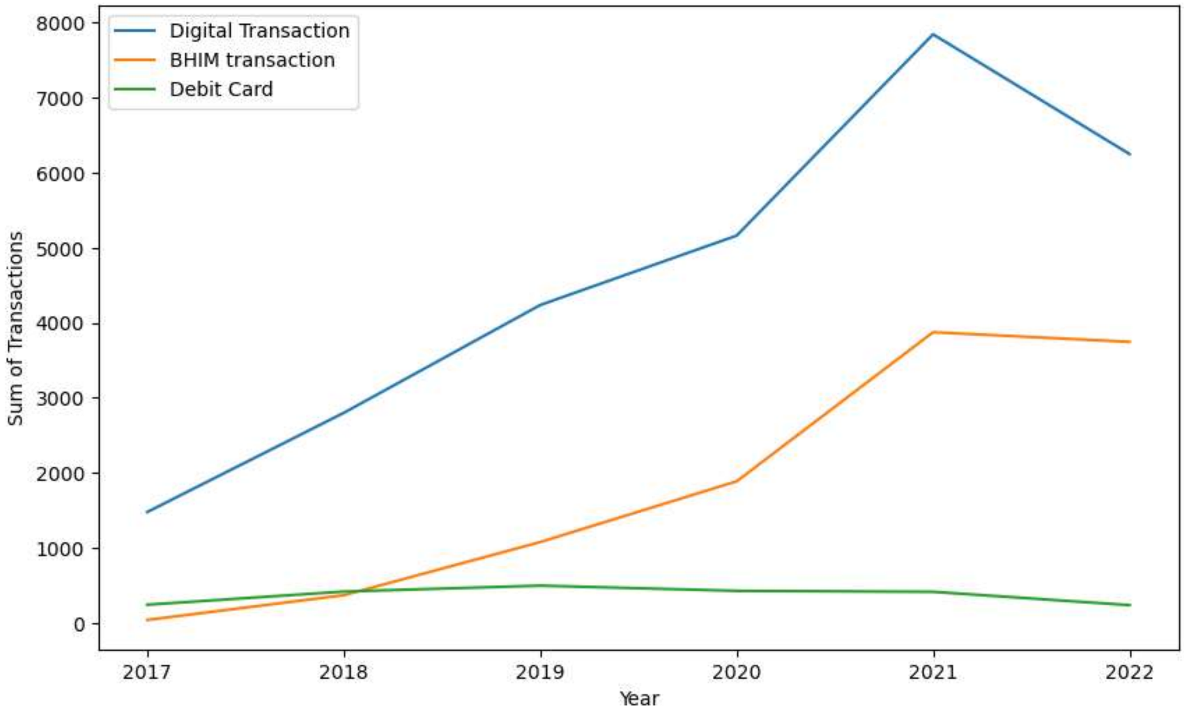
Out[7]:

```
ROWID                   0
Name of the ministry    0
Name of the Project     0
SourceMonth             0
SourceYear              0
Digital Transaction     0
BHIM transaction        0
Debit Card              0
Country                 0
YearCode                0
Year                    0
MonthCode               0
Month                   0
dtype: int64
```

In [32]:

```
 # Use groupby and sum to calculate the sum of transactions for each type and year
grouped_df = df.groupby('Year').agg({
    'Digital Transaction ': 'sum',
    'BHIM transaction': 'sum',
    'Debit Card': 'sum'
}).reset_index()

# Display the grouped DataFrame
print(grouped_df)

# Plot the data
plt.figure(figsize=(10, 6))
plt.plot(grouped_df['Year'], grouped_df['Digital Transaction '], label='Digital Tra
plt.plot(grouped_df['Year'], grouped_df['BHIM transaction'], label='BHIM transactic
plt.plot(grouped_df['Year'], grouped_df['Debit Card'], label='Debit Card')
plt.xlabel('Year')
plt.ylabel('Sum of Transactions')
plt.legend()
plt.show()
```

```
   Year  Digital Transaction   BHIM transaction  Debit Card
0  2017              1479.28             41.23      244.11
1  2018              2800.00            370.63      419.18
2  2019              4236.24           1078.63      497.88
3  2020              5161.08           1887.98      428.95
4  2021              7842.52           3874.33      415.84
5  2022              6248.16           3745.64      238.40
```



In [37]:

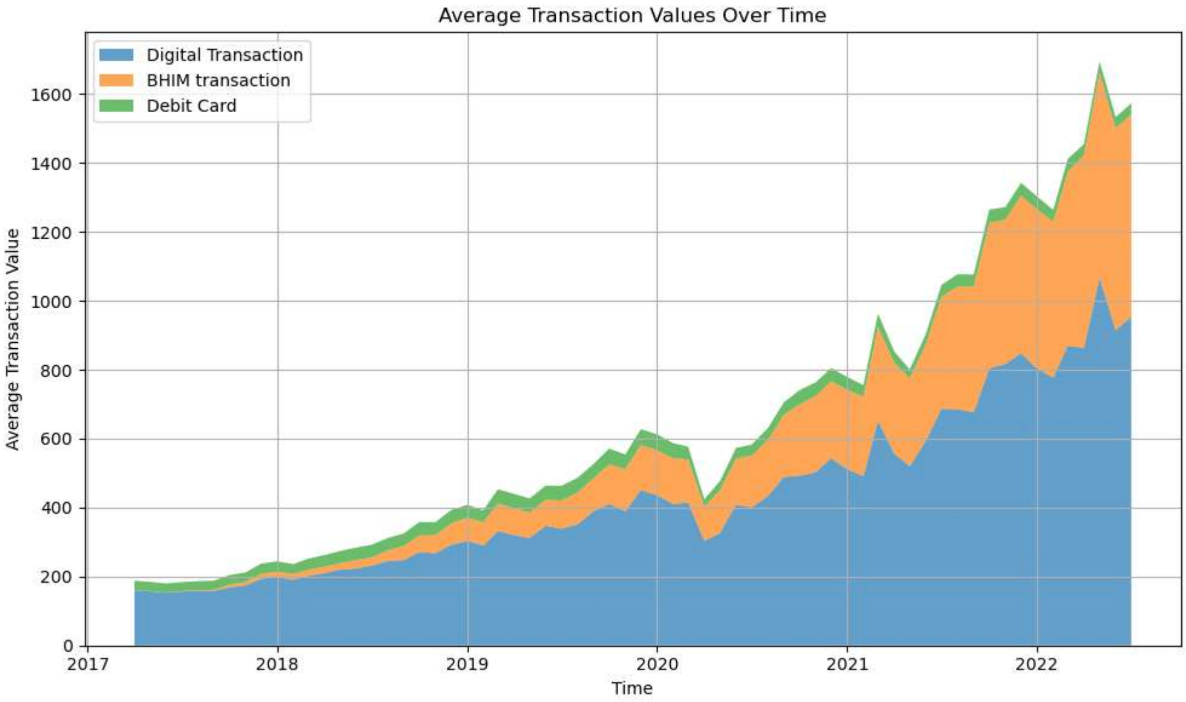
```
 # Sort the DataFrame by 'Digital Transaction ' in descending order
sorted_digital_df = df.sort_values(by='Digital Transaction ', ascending=False)

# Select the top-performing months for Digital Transaction (e.g., top 5)
top_digital_months = sorted_digital_df.head(5)

# Sort the DataFrame by 'BHIM transaction' in descending order
sorted_bhim_df = df.sort_values(by='BHIM transaction', ascending=False)

# Select the top-performing months for BHIM Transaction (e.g., top 5)
top_bhim_months = sorted_bhim_df.head(5)

# Sort the DataFrame by 'Debit Card' in descending order
```

```
sorted_debit_df = df.sort_values(by='Debit Card', ascending=False)

# Select the top-performing months for Debit Card (e.g., top 5)
top_debit_months = sorted_debit_df.head(5)

# Display the top-performing months for each type
print("Top Performing Months - Digital Transaction:")
print(top_digital_months[['MonthCode', 'Month', 'Digital Transaction ']])
print("\nTop Performing Months - BHIM Transaction:")
print(top_bhim_months[['MonthCode', 'Month', 'BHIM transaction']])
print("\nTop Performing Months - Debit Card:")
print(top_debit_months[['MonthCode', 'Month', 'Debit Card']])
```

```
Top Performing Months - Digital Transaction:
    MonthCode        Month  Digital Transaction 
19 2022-05-01    May, 2022              1066.31
22 2022-07-01   July, 2022               953.91
16 2022-06-01   June, 2022               914.38
41 2022-03-01  March, 2022               869.69
18 2022-04-01  April, 2022               863.65

Top Performing Months - BHIM Transaction:
    MonthCode        Month  BHIM transaction
19 2022-05-01    May, 2022            595.53
22 2022-07-01   July, 2022            586.38
16 2022-06-01   June, 2022            586.27
18 2022-04-01  April, 2022            558.30
41 2022-03-01  March, 2022            504.71

Top Performing Months - Debit Card:
    MonthCode           Month  Debit Card
63 2020-01-01   January, 2020       46.37
62 2019-12-01  December, 2019       45.79
61 2019-10-01   October, 2019       45.50
60 2020-02-01  February, 2020       43.81
59 2019-11-01  November, 2019       43.00
```

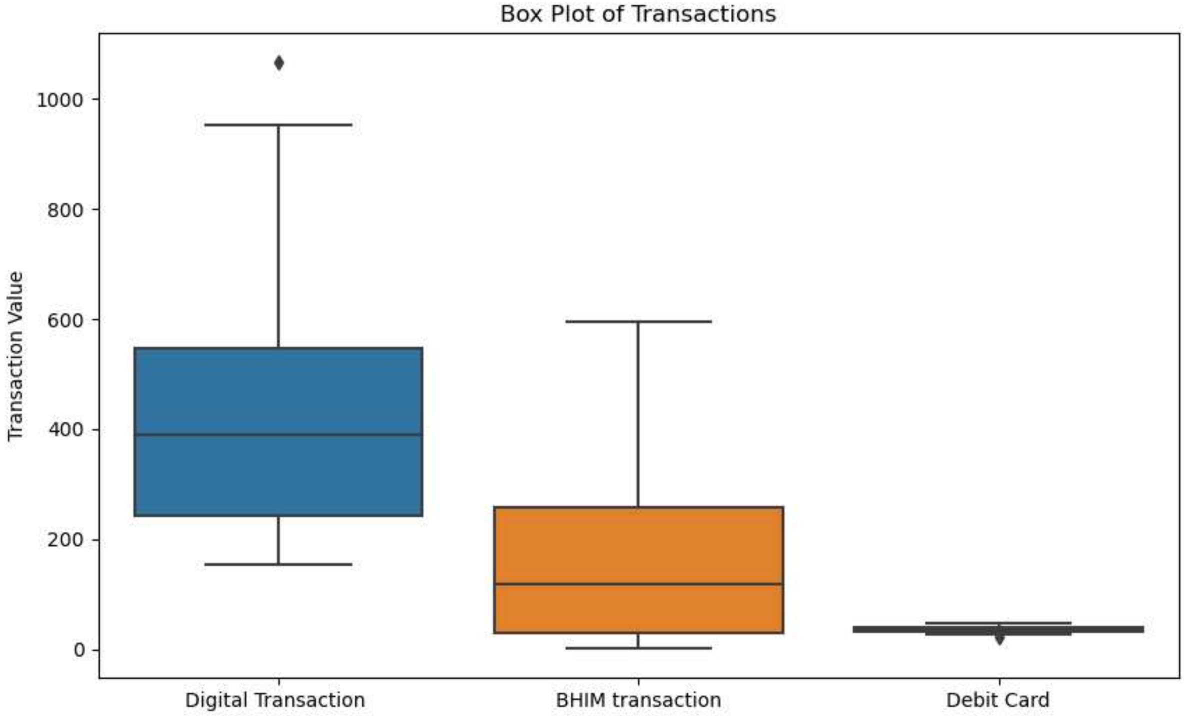
In [44]:

```
# Convert 'MonthCode' to datetime type
df['MonthCode'] = pd.to_datetime(df['MonthCode'], format='%Y%m')

# Group by 'MonthCode' and calculate the average for each type of transaction
average_df = df.groupby('MonthCode').agg({
    'Digital Transaction ': 'mean',
    'BHIM transaction': 'mean',
    'Debit Card': 'mean'
}).reset_index()


# Plotting as a stacked area chart
plt.figure(figsize=(10, 6))

plt.stackplot(average_df['MonthCode'], average_df['Digital Transaction '], average_
              labels=['Digital Transaction', 'BHIM transaction', 'Debit Card'], alp

# Formatting
plt.xlabel('Time')
plt.ylabel('Average Transaction Value')
plt.title('Average Transaction Values Over Time')
plt.legend(loc='upper left')
plt.grid(True)
plt.tight_layout()
plt.show()
```



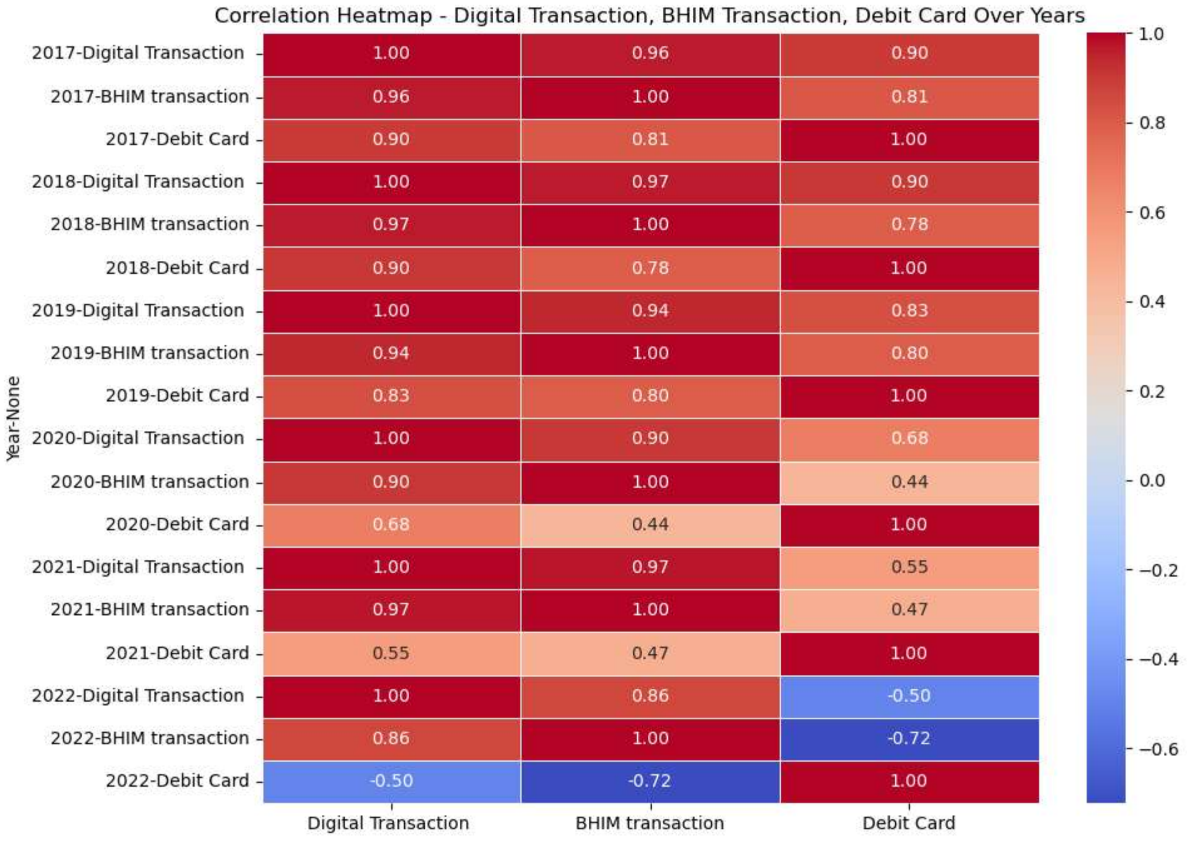
In [46]:

```
# Box Plot Showing the distribution of transactions
plt.figure(figsize=(10, 6))
sns.boxplot(data=df[['Digital Transaction ', 'BHIM transaction', 'Debit Card']])
plt.title('Box Plot of Transactions')
plt.ylabel('Transaction Value')
plt.show()
```



In [48]:

```
# Select relevant columns for correlation analysis
Types_of_transaction= ['Digital Transaction ', 'BHIM transaction', 'Debit Card', 'Y
df_selected = df[Types_of_transaction]

# Calculate the correlation matrix
correlation_matrix = df_selected.groupby('Year').corr()

# Create a heatmap
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f", linewidths=
plt.title('Correlation Heatmap - Digital Transaction, BHIM Transaction, Debit Card
plt.show()
```



In [ ]: