```
## Credit Card Spending In India ##
```

```
# 📦 1. Import Libraries
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

```
# 📥 2. Load the Dataset
df = pd.read_csv("Credit Card Spending in India.csv")
df
```

Out[115]:

	index	City	Date	Card Type	Exp Type	Gender	Amount
0	0.0	Delhi, India	29-Oct-14	Gold	Bills	F	82475.0
1	1.0	Greater Mumbai, India	22-Aug-14	Platinum	Bills	F	32555.0
2	2.0	Bengaluru, India	27-Aug-14	Silver	Bills	F	101738.0
3	3.0	Greater Mumbai, India	12-Apr-14	Signature	Bills	F	123424.0
4	4.0	Bengaluru, India	5-May-15	Gold	Bills	F	171574.0
...	...	...	...	...	...	...	...
26047	NaN	NaN	NaN	NaN	NaN	NaN	NaN
26048	NaN	NaN	NaN	NaN	NaN	NaN	NaN
26049	NaN	NaN	NaN	NaN	NaN	NaN	NaN
26050	NaN	NaN	NaN	NaN	NaN	NaN	NaN
26051	NaN	NaN	NaN	NaN	NaN	NaN	NaN

26052 rows × 7 columns

```
# 🧹 3. Data Cleaning

# (1) Convert date column explicitly
df['Date'] = pd.to_datetime(df['Date'], format="%d-%b-%y", errors='coerce')

# (2) Drop rows with missing date or amount
df.dropna(subset=['Date', 'Amount'], inplace=True)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 101 entries, 0 to 100
Data columns (total 6 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   City       101 non-null    object
 1   Date       101 non-null    datetime64[ns]
 2   Card Type  101 non-null    object
 3   Exp Type   101 non-null    object
 4   Gender     101 non-null    object
 5   Amount     101 non-null    float64
dtypes: datetime64[ns](1), float64(1), object(4)
memory usage: 5.5+ KB
```

```
df.describe()
```

Out[123]:

	Date	Amount
count	101	101.000000
mean	2014-07-12 15:12:28.514851584	517599.831683
min	2013-10-04 00:00:00	18026.000000
25%	2014-02-27 00:00:00	302834.000000
50%	2014-06-26 00:00:00	525977.000000
75%	2015-01-08 00:00:00	722587.000000
max	2015-05-13 00:00:00	998077.000000
std	NaN	261440.499690

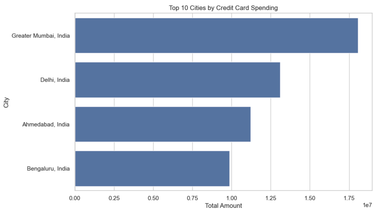
```
df.head()
```

Out[125]:

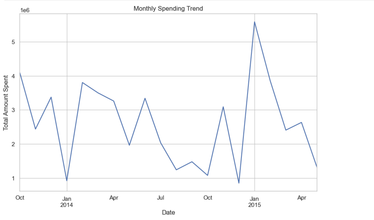
	City	Date	Card Type	Exp Type	Gender	Amount
0	Delhi, India	2014-10-29	Gold	Bills	F	82475.0
1	Greater Mumbai, India	2014-08-22	Platinum	Bills	F	32555.0
2	Bengaluru, India	2014-08-27	Silver	Bills	F	101738.0
3	Greater Mumbai, India	2014-04-12	Signature	Bills	F	123424.0
4	Bengaluru, India	2015-05-05	Gold	Bills	F	171574.0

```
# 🎨 Set Visualization Style
sns.set(style="whitegrid")
plt.rcParams['figure.figsize'] = (10, 6)
```

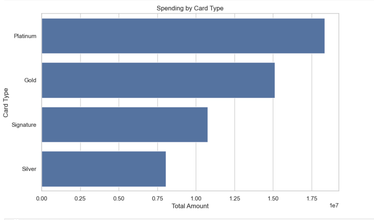
```
# 🏙 Top 10 Cities by Total Spending
top_cities = df.groupby('City')['Amount'].sum().sort_values(ascending=False).head(10)
sns.barplot(x=top_cities.values, y=top_cities.index)
plt.title("Top 10 Cities by Credit Card Spending")
plt.xlabel("Total Amount")
plt.ylabel("City")
plt.show()
```




```
# 📈 Monthly Spending Trend
# 📈 Monthly Spending Trend
monthly_trend = df.resample('ME', on='Date')['Amount'].sum()
monthly_trend.plot()
plt.title("Monthly Spending Trend")
plt.xlabel("Date")
plt.ylabel("Total Amount Spent")
plt.grid(True)
plt.show()
```

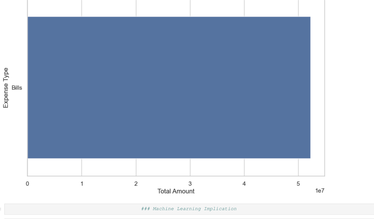



```
# 💳 Spending by Card Type
card_type_spending = df.groupby('Card Type')['Amount'].sum().sort_values(ascending=False)
sns.barplot(x=card_type_spending.values, y=card_type_spending.index)
plt.title("Spending by Card Type")
plt.xlabel("Total Amount")
plt.ylabel("Card Type")
plt.show()
```




```
# 🚻 Spending by Gender
gender_spending = df.groupby('Gender')['Amount'].sum()
sns.barplot(x=gender_spending.index, y=gender_spending.values)
plt.title("Spending by Gender")
plt.xlabel("Gender")
plt.ylabel("Total Amount")
plt.show()
```


```
# 🧾 Spending by Expense Type
exp_type_spending = df.groupby('Exp Type')['Amount'].sum().sort_values(ascending=False)
sns.barplot(x=exp_type_spending.values, y=exp_type_spending.index)
plt.title("Spending by Expense Type")
plt.xlabel("Total Amount")
plt.ylabel("Expense Type")
plt.show()
```




```
### Machine Learning Implication
```

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_absolute_error, r2_score
```

```
# 📅 Feature Engineering: Extract Month and Year
df['Year'] = df['Date'].dt.year
df['Month'] = df['Date'].dt.month
```

```
# 🎯 Select features and target
features = ['City', 'Card Type', 'Exp Type', 'Gender', 'Year', 'Month']
target = 'Amount'

X = df[features]
y = df[target]
```

```
# One-hot encode categorical variables
categorical_features = ['City', 'Card Type', 'Exp Type', 'Gender']
numerical_features = ['Year', 'Month']

preprocessor = ColumnTransformer(
    transformers=[
        ('cat', OneHotEncoder(handle_unknown='ignore'), categorical_features)
    ],
    remainder='passthrough'  # Keep numerical columns
)
```

```
# Create ML Pipeline
pipeline = Pipeline(steps=[
    ('preprocess', preprocessor),
    ('model', RandomForestRegressor(n_estimators=100, random_state=42))
])
```

```
# Split data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Train model
pipeline.fit(X_train, y_train)

# Predict
y_pred = pipeline.predict(X_test)
```

```
# Evaluation metrics
mae = mean_absolute_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f"🔮 Model Evaluation:")
print(f"Mean Absolute Error (MAE): {round(mae, 2)}")
print(f"R² Score:", round(r2, 4))
```

```
🔮 Model Evaluation:
Mean Absolute Error (MAE): 277969.66
R² Score: -0.8137
```

```
# Example: Predict for new input
sample = pd.DataFrame([{
    'City': 'Delhi, India',
    'Card Type': 'Gold',
    'Exp Type': 'Bills',
    'Gender': 'F',
    'Year': 2024,
    'Month': 7
}])
```

```
predicted_amount = pipeline.predict(sample)
print("Predicted Amount:", predicted_amount[0])
```

```
Predicted Amount: 620542.7466666666
```

In [47]:
```
## End Project ##
```

In [ ]:
```

```