Day23_EDA_7_Tech_On_Dataset.ipynb

June 13, 2025

Practical EDA: Applying 7 Core Techniques on a Real Dataset

Today, we will apply 7 essential EDA techniques to a small dataset.
This process works whether your dataset has 7 rows or 7 million!

Recap from Day 22

We learned the theory behind:

1. Variable Identification
2. Univariate Analysis
3. Bivariate Analysis
4. Outlier Detection
5. Missing Value Treatment
6. Variable Transformation
7. Variable Creation

Dataset Preview

Name	Domain	Age	Location	Salary	Exp
Mike	`Datascience#$`	34 years	Mumbai	`5^00#0`	2+
Teddy^	Testing	45' yr	Bangalore	`10%%000`	<3
Umar#r	`Dataanalyst^^#`	NaN	NaN	`1$5%000`	4> yrs
Jane	`Ana^^lytics`	NaN	Hyderbad	`2000^0`	NaN
Uttam*	Statistics	67-yr	NaN	`30000-`	5+ year
Kim	NLP	55yr	Delhi	`6000^$0`	10+

Important Note

Even though this dataset has only 6 rows, the same EDA techniques we apply here can be used on datasets with **thousands or even millions of rows**.
EDA is not about size — it's about understanding, cleaning, and preparing your data for analysis and machine learning.

Plan of Action for Today

We will apply each of the 7 EDA techniques to this dataset in order to: - Clean inconsistent and messy values
- Handle missing data
- Standardize column types
- Explore data patterns
- Prepare it for machine learning models

[1]:
```
# Import Libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

[2]:
```
# Load the Excel file
emp = pd.read_excel(r'C:\Users\aksha\OneDrive\Desktop\Dataset\EDA\Rawdata.xlsx')
```

[3]:
```
emp
```

[3]:

	Name	Domain	Age	Location	Salary	Exp
0	Mike	Datascience#$	34 years	Mumbai	5^00#0	2+
1	Teddy^	Testing	45' yr	Bangalore	10%%000	<3
2	Uma#r	Dataanalyst^^#	NaN	NaN	1$5%000	4> yrs
3	Jane	Ana^^lytics	NaN	Hyderbad	2000^0	NaN
4	Uttam*	Statistics	67-yr	NaN	30000-	5+ year
5	Kim	NLP	55yr	Delhi	6000^$0	10+

[4]:
```
# #  Load Raw Data Manually
# data = {
#     'Name': ['Mike', 'Teddy^', 'Umar#r', 'Jane', 'Uttam*', 'Kim'],
#     'Domain': ['Datascience#$', 'Testing', 'Dataanalyst^^#', 'Ana^^lytics',␣
 ↪'Statistics', 'NLP'],
#     'Age': ['34 years', "45' yr", np.nan, np.nan, '67-yr', '55yr'],
#     'Location': ['Mumbai', 'Bangalore', np.nan, 'Hyderbad', np.nan, 'Delhi'],
#     'Salary': ['5^00#0', '10%%000', '1$5%000', '2000^0', '30000-', '6000^$0'],
#     'Exp': ['2+', '<3', '4> yrs', np.nan, '5+ year', '10+']
# }

# emp = pd.DataFrame(data)
# emp
```

1 Basic Data Inspection

[5]:
```
emp.head()  # Displays the first 5 rows
```

[5]:

	Name	Domain	Age	Location	Salary	Exp
0	Mike	Datascience#$	34 years	Mumbai	5^00#0	2+
1	Teddy^	Testing	45' yr	Bangalore	10%%000	<3
2	Uma#r	Dataanalyst^^#	NaN	NaN	1$5%000	4> yrs
3	Jane	Ana^^lytics	NaN	Hyderbad	2000^0	NaN
4	Uttam*	Statistics	67-yr	NaN	30000-	5+ year

[6]:
```
emp.tail()  # Displays the last 5 rows
```

[6]:

	Name	Domain	Age	Location	Salary	Exp
1	Teddy^	Testing	45' yr	Bangalore	10%%000	<3
2	Uma#r	Dataanalyst^^#	NaN	NaN	1$5%000	4> yrs
3	Jane	Ana^^lytics	NaN	Hyderbad	2000^0	NaN
4	Uttam*	Statistics	67-yr	NaN	30000-	5+ year
5	Kim	NLP	55yr	Delhi	6000^$0	10+

[7]:
```
emp.shape        # Returns (rows, columns)
```

[7]:
```
(6, 6)
```

[8]:
```
emp.columns      # Lists all column names
```

[8]:
```
Index(['Name', 'Domain', 'Age', 'Location', 'Salary', 'Exp'], dtype='object')
```

2 Dataset Summary Information

[9]:
```
# Dataset Summary Information
emp.info()       # Overview of data types, non-null counts, and memory usage
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6 entries, 0 to 5
Data columns (total 6 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   Name      6 non-null      object
 1   Domain    6 non-null      object
 2   Age       4 non-null      object
 3   Location  4 non-null      object
 4   Salary    6 non-null      object
 5   Exp       5 non-null      object
dtypes: object(6)
memory usage: 420.0+ bytes
```

3 Missing Value Check

[10]:
```
# Missing Value Check
emp.isnull()     # Returns a DataFrame showing True for missing cells
```

[10]:

	Name	Domain	Age	Location	Salary	Exp
0	False	False	False	False	False	False
1	False	False	False	False	False	False
2	False	False	True	True	False	False
3	False	False	True	False	False	True
4	False	False	False	True	False	False
5	False	False	False	False	False	False

[11]:
```
emp.isna()       # Same as isnull(), both can be used interchangeably
```

[11]:

	Name	Domain	Age	Location	Salary	Exp
0	False	False	False	False	False	False
1	False	False	False	False	False	False
2	False	False	True	True	False	False
3	False	False	True	False	False	True
4	False	False	False	True	False	False
5	False	False	False	False	False	False

[12]:
```
emp.isnull().sum()    # Total number of missing values in each column
```

[12]:
```
Name        0
Domain      0
Age         2
Location    2
Salary      0
Exp         1
dtype: int64
```

4 EDA Technique

4.1 Variable Transformation: Cleaned symbols, converted types

[13]:
```
# Remove unwanted characters
# Remove all non-word characters from the 'Name' column
emp['Name'] = emp['Name'].str.replace(r'\W', '', regex=True)
emp
```

[13]:

	Name	Domain	Age	Location	Salary	Exp
0	Mike	Datascience#$	34 years	Mumbai	5^00#0	2+
1	Teddy	Testing	45' yr	Bangalore	10%%000	<3
2	Umar	Dataanalyst^^#	NaN	NaN	1$5%000	4> yrs
3	Jane	Ana^^lytics	NaN	Hyderbad	2000^0	NaN

```
4  Uttam      Statistics     67-yr        NaN   30000-  5+ year
5    Kim             NLP      55yr      Delhi  6000^$0      10+
```

[14]:
```
# Do same for all columns
emp['Domain'] = emp['Domain'].str.replace(r'\W', '', regex=True)
emp['Age'] = emp['Age'].str.replace(r'\W', '', regex=True)
emp['Salary'] = emp['Salary'].str.replace(r'\W', '', regex=True)
emp['Location'] = emp['Location'].str.replace(r'\W', '', regex=True)
emp
```

[14]:

	Name	Domain	Age	Location	Salary	Exp
0	Mike	Datascience	34years	Mumbai	5000	2+
1	Teddy	Testing	45yr	Bangalore	10000	<3
2	Umar	Dataanalyst	NaN	NaN	15000	4> yrs
3	Jane	Analytics	NaN	Hyderbad	20000	NaN
4	Uttam	Statistics	67yr	NaN	30000	5+ year
5	Kim	NLP	55yr	Delhi	60000	10+

[15]:
```
# Extract digits from 'Age' and 'Exp'
emp['Age'] = emp['Age'].str.extract(r'(\d+)')
emp['Exp'] = emp['Exp'].str.extract(r'(\d+)')
emp
```

[15]:

	Name	Domain	Age	Location	Salary	Exp
0	Mike	Datascience	34	Mumbai	5000	2
1	Teddy	Testing	45	Bangalore	10000	3
2	Umar	Dataanalyst	NaN	NaN	15000	4
3	Jane	Analytics	NaN	Hyderbad	20000	NaN
4	Uttam	Statistics	67	NaN	30000	5
5	Kim	NLP	55	Delhi	60000	10

4.2 Missing Value Treatment: Used mean and mode

[16]:
```
emp
```

[16]:

	Name	Domain	Age	Location	Salary	Exp
0	Mike	Datascience	34	Mumbai	5000	2
1	Teddy	Testing	45	Bangalore	10000	3
2	Umar	Dataanalyst	NaN	NaN	15000	4
3	Jane	Analytics	NaN	Hyderbad	20000	NaN
4	Uttam	Statistics	67	NaN	30000	5
5	Kim	NLP	55	Delhi	60000	10

[17]:
```
# Convert Age and Exp to numeric
emp['Age'] = pd.to_numeric(emp['Age'])
emp['Exp'] = pd.to_numeric(emp['Exp'])
```

[18]:
```
# fill missing values with mean (numeric values)
emp['Age'] = emp['Age'].fillna(emp['Age'].mean())
emp['Exp'] = emp['Exp'].fillna(emp['Exp'].mean())
emp
```

[18]:

	Name	Domain	Age	Location	Salary	Exp
0	Mike	Datascience	34.00	Mumbai	5000	2.0
1	Teddy	Testing	45.00	Bangalore	10000	3.0
2	Umar	Dataanalyst	50.25	NaN	15000	4.0
3	Jane	Analytics	50.25	Hyderbad	20000	4.8
4	Uttam	Statistics	67.00	NaN	30000	5.0
5	Kim	NLP	55.00	Delhi	60000	10.0

[19]:
```
# Fill categorical nulls with mode
emp['Location'] = emp['Location'].fillna(emp['Location'].mode()[0])
emp
```

[19]:

	Name	Domain	Age	Location	Salary	Exp
0	Mike	Datascience	34.00	Mumbai	5000	2.0
1	Teddy	Testing	45.00	Bangalore	10000	3.0
2	Umar	Dataanalyst	50.25	Bangalore	15000	4.0
3	Jane	Analytics	50.25	Hyderbad	20000	4.8
4	Uttam	Statistics	67.00	Bangalore	30000	5.0
5	Kim	NLP	55.00	Delhi	60000	10.0

Convert Data Types

Before performing data type conversions (like `.astype(int)`), it is essential to handle missing values.

- You must either:
  - Fill missing values using `.fillna()`
  - Or drop them using `.dropna()`
- Otherwise, pandas will raise errors like `IntCastingNaNError` because types like `int` and `category` **do not support NaN**.

[20]:
```
emp['Age'] = emp['Age'].astype(int)
emp['Exp'] = emp['Exp'].astype(int)
emp['Salary'] = pd.to_numeric(emp['Salary'])

emp['Name'] = emp['Name'].astype('category')
emp['Domain'] = emp['Domain'].astype('category')
emp['Location'] = emp['Location'].astype('category')

emp.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6 entries, 0 to 5
```

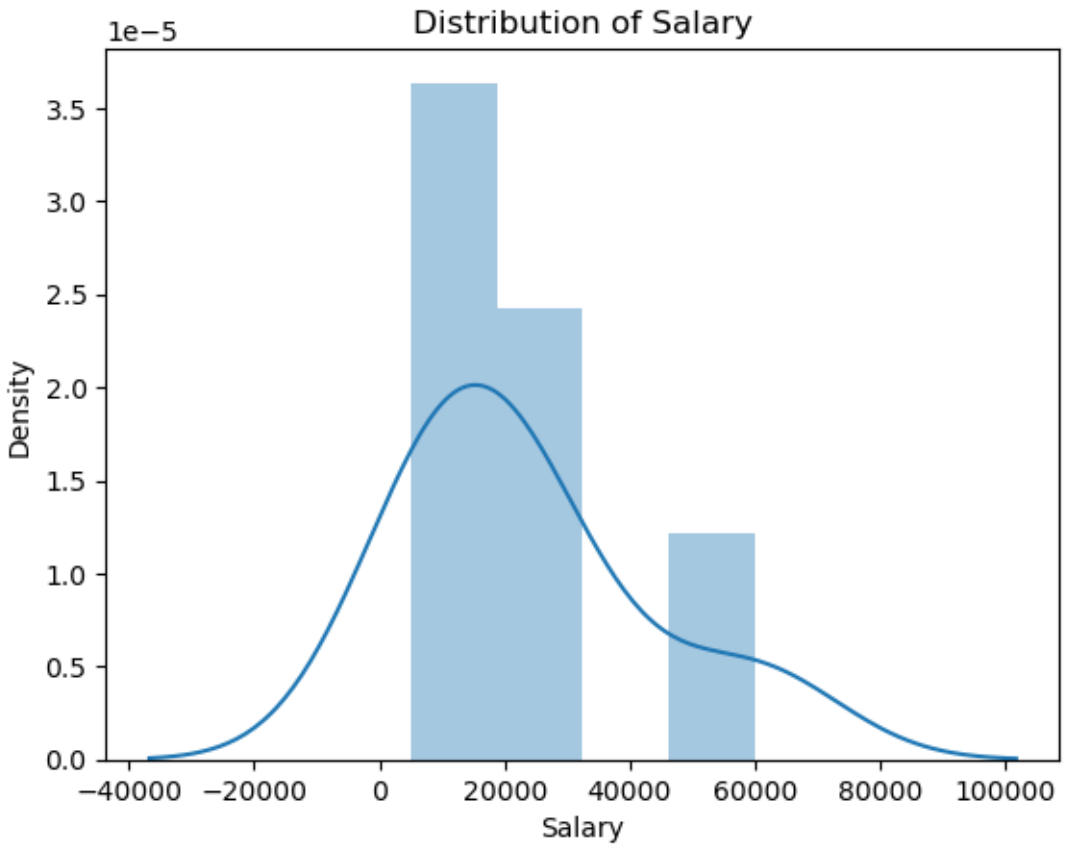
```
Data columns (total 6 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   Name      6 non-null      category
 1   Domain    6 non-null      category
 2   Age       6 non-null      int32
 3   Location  6 non-null      category
 4   Salary    6 non-null      int64
 5   Exp       6 non-null      int32
dtypes: category(3), int32(2), int64(1)
memory usage: 890.0 bytes
```

[21]:
```
emp.to_csv('Clean_data.csv', index=False)
# Check your current working directory using:
import os
os.getcwd()
```

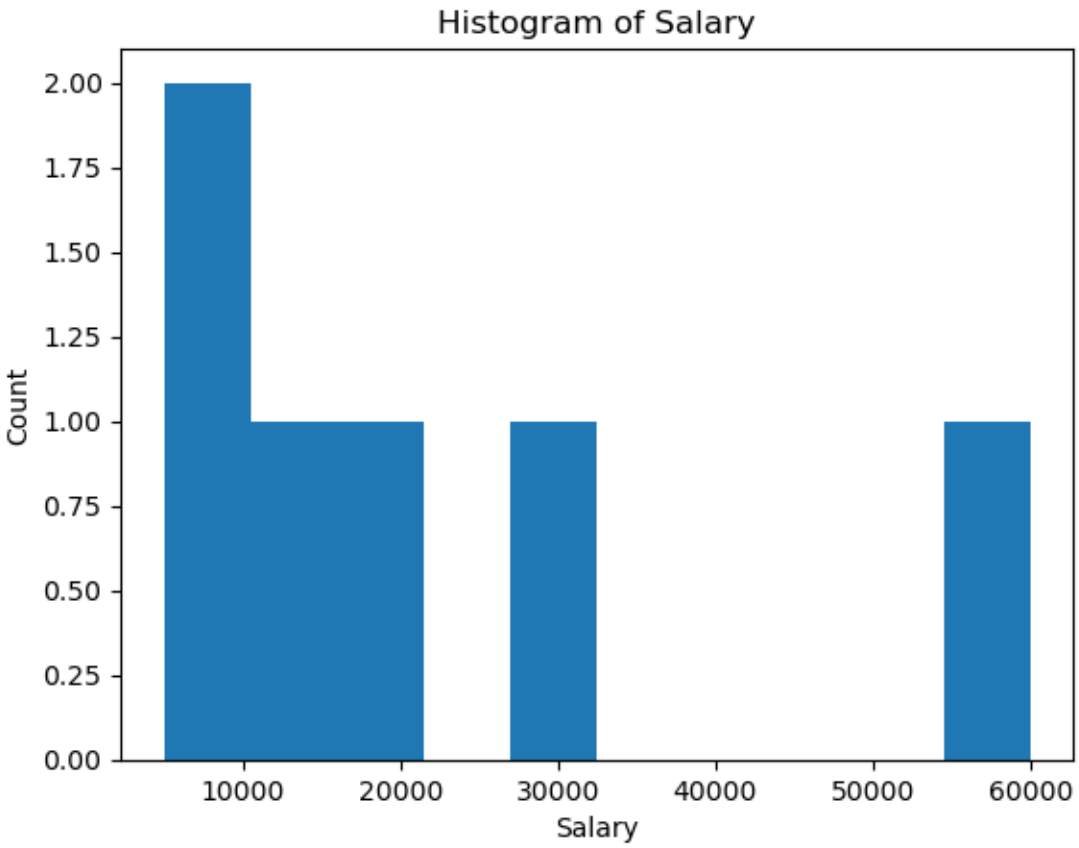
[21]:
```
'C:\\Users\\aksha\\OneDrive\\Desktop\\Full stack Data Science course\\GITHUB
Uploads\\4_EDA_Exploratory_Data_Analysis'
```

4.3 Univariate Analysis: Distplots, histograms

[22]:
```
# Univariate Plot
sns.distplot(emp['Salary'])
plt.title("Distribution of Salary")
plt.show()
```

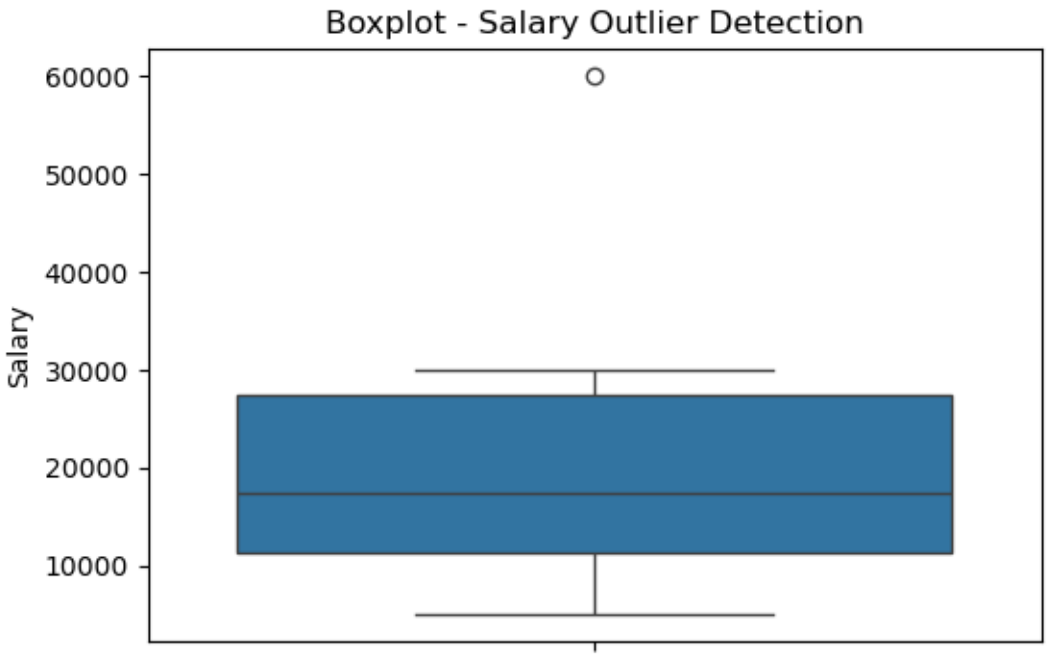


[23]:
```
# Histogram
plt.hist(emp['Salary'])
plt.title("Histogram of Salary")
plt.xlabel("Salary")
plt.ylabel("Count")
plt.show()
```

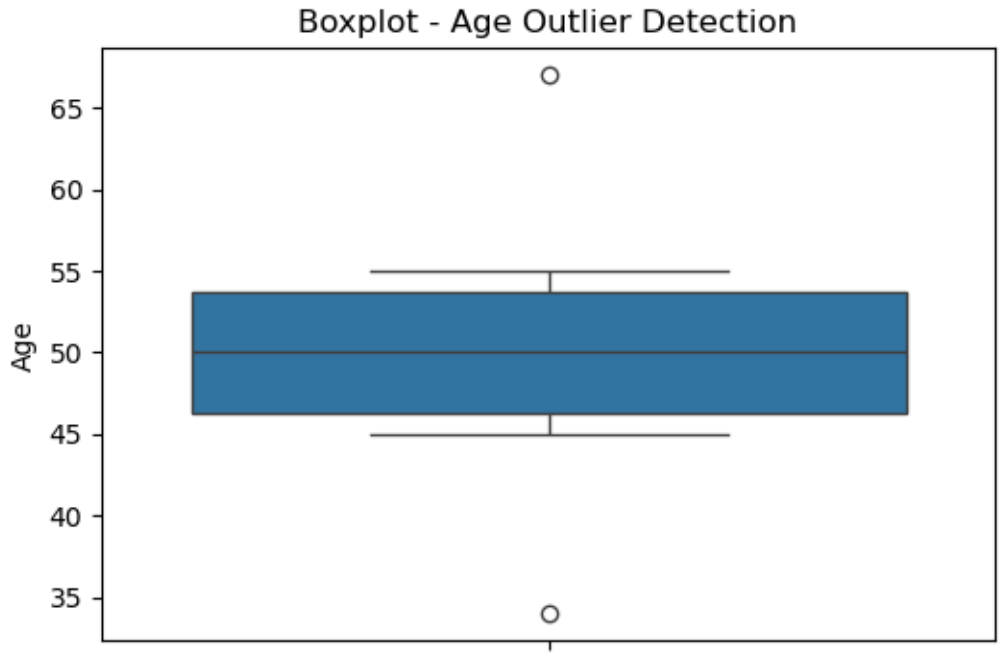


4.4 Outlier Detection: Visually via plots

[24]:
```
plt.figure(figsize=(6, 4))
sns.boxplot(emp['Salary'])
plt.title("Boxplot - Salary Outlier Detection")
plt.show()
```



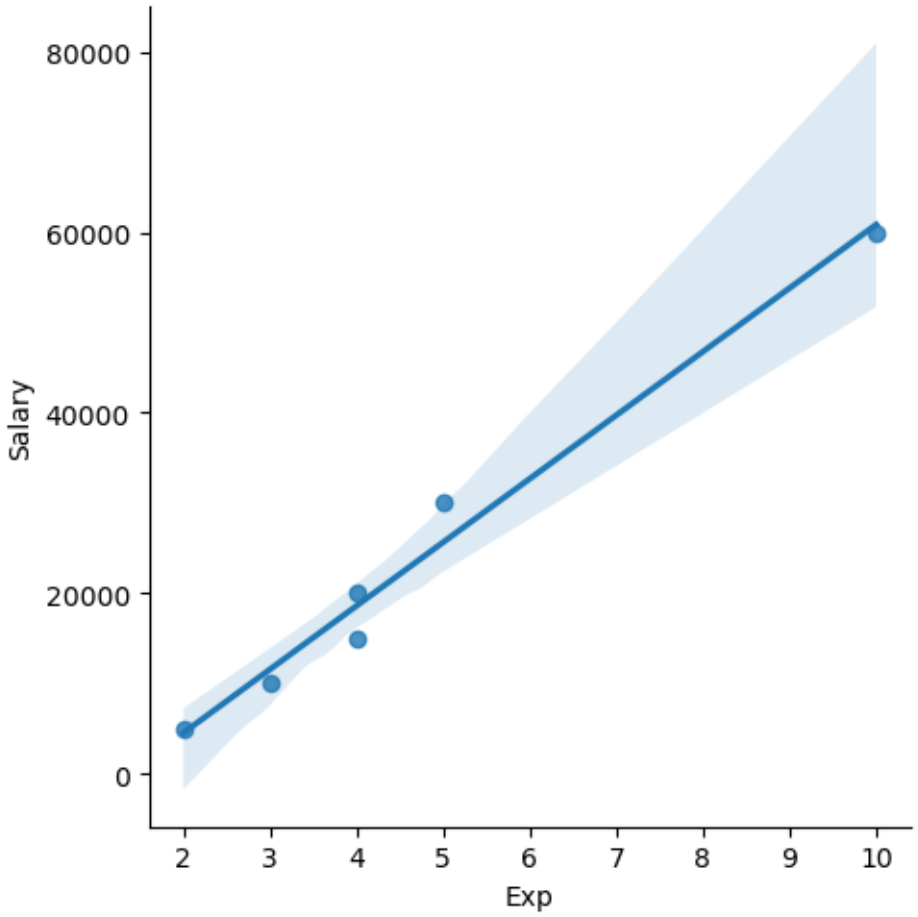
[25]:
```
plt.figure(figsize=(6, 4))
sns.boxplot(emp['Age'])
plt.title("Boxplot - Age Outlier Detection")
plt.show()
```



4.5 Bivariate Analysis: Regression plots with seaborn

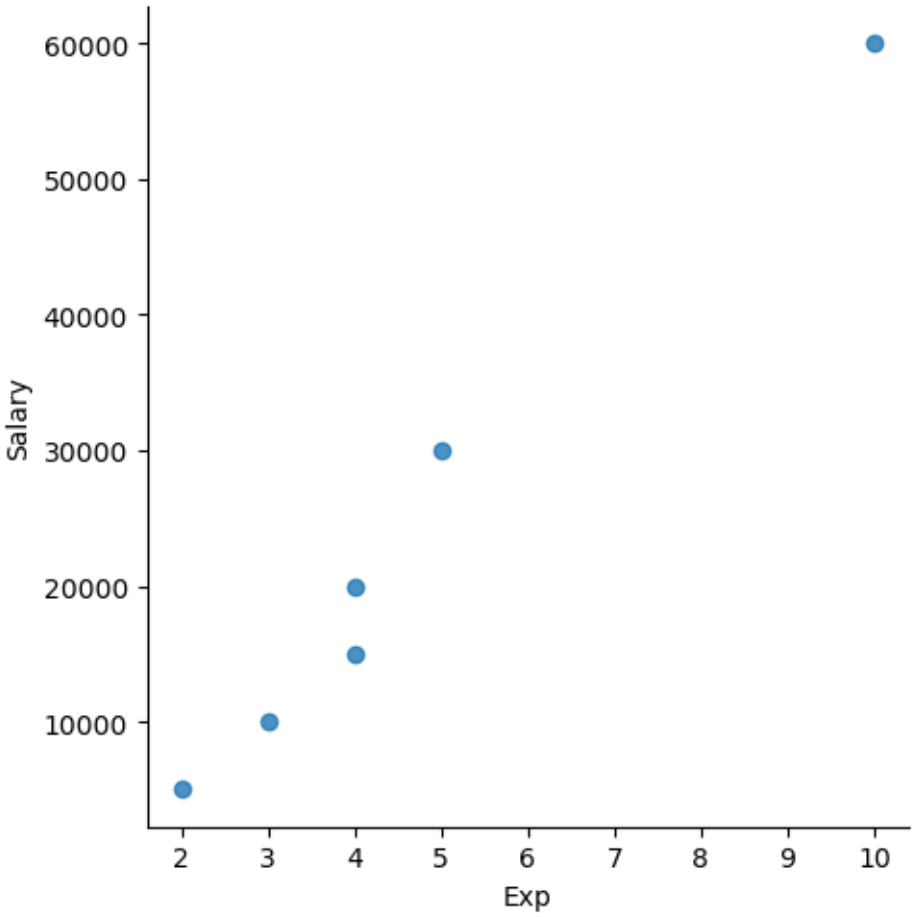
[26]:
```
# Bivariate - Regression line
sns.lmplot(x='Exp', y='Salary', data=emp)
```

[26]: <seaborn.axisgrid.FacetGrid at 0x24478189310>



[27]:
```
# Bivariate - Without regression
sns.lmplot(x='Exp', y='Salary', data=emp, fit_reg=False)
```

[27]: <seaborn.axisgrid.FacetGrid at 0x2447826be60>



4.6 Variable Identification: Selected X_iv and y_dv

[28]:
```
# Slicing and Indexing before Variable Identification
df = emp.copy()
df[:]       # All rows
```

[28]:

	Name	Domain	Age	Location	Salary	Exp
0	Mike	Datascience	34	Mumbai	5000	2
1	Teddy	Testing	45	Bangalore	10000	3
2	Umar	Dataanalyst	50	Bangalore	15000	4
3	Jane	Analytics	50	Hyderbad	20000	4
4	Uttam	Statistics	67	Bangalore	30000	5
5	Kim	NLP	55	Delhi	60000	10

[29]:
```
df[0:6:2]     # Every second row from first 6
```

[29]:

	Name	Domain	Age	Location	Salary	Exp
0	Mike	Datascience	34	Mumbai	5000	2

```
2   Umar  Dataanalyst   50  Bangalore   15000    4
4  Uttam   Statistics   67  Bangalore   30000    5
```

[30]:
```
df[::-1]     # Reverse order of rows
```

[30]:
```
    Name       Domain  Age   Location  Salary  Exp
5    Kim          NLP   55      Delhi   60000   10
4  Uttam   Statistics   67  Bangalore   30000    5
3   Jane    Analytics   50   Hyderbad   20000    4
2   Umar  Dataanalyst   50  Bangalore   15000    4
1  Teddy      Testing   45  Bangalore   10000    3
0   Mike  Datascience   34     Mumbai    5000    2
```

[31]:
```
# Splitting Features (X_iv) and Target (y_dv)
X_iv = emp[['Name', 'Domain', 'Age', 'Location', 'Exp']] # Independent variables
y_dv = emp[['Salary']]  # Dependent variable
```

[32]:
```
X_iv
```

[32]:
```
    Name       Domain  Age   Location  Exp
0   Mike  Datascience   34     Mumbai    2
1  Teddy      Testing   45  Bangalore    3
2   Umar  Dataanalyst   50  Bangalore    4
3   Jane    Analytics   50   Hyderbad    4
4  Uttam   Statistics   67  Bangalore    5
5    Kim          NLP   55      Delhi   10
```

[33]:
```
y_dv
```

[33]:
```
   Salary
0    5000
1   10000
2   15000
3   20000
4   30000
5   60000
```

4.7 Variable Creation: One-hot encoded categorical variables

[34]:
```
# One-hot encoding
imputation = pd.get_dummies(df)
```

[35]:
```
imputation
```

[35]:
```
   Age  Salary  Exp  Name_Jane  Name_Kim  Name_Mike  Name_Teddy  Name_Umar  \
0   34    5000    2      False     False       True       False      False
1   45   10000    3      False     False      False        True      False
2   50   15000    4      False     False      False       False       True
```

```
3   50   20000    4       True     False      False       False      False
4   67   30000    5      False     False      False       False      False
5   55   60000   10      False      True      False       False      False

   Name_Uttam  Domain_Analytics  Domain_Dataanalyst  Domain_Datascience  \
0       False             False               False                True
1       False             False               False               False
2       False             False                True               False
3       False              True               False               False
4        True             False               False               False
5       False             False               False               False

   Domain_NLP  Domain_Statistics  Domain_Testing  Location_Bangalore  \
0       False              False           False               False
1       False              False            True                True
2       False              False           False                True
3       False              False           False               False
4       False               True           False                True
5        True              False           False               False

   Location_Delhi  Location_Hyderbad  Location_Mumbai
0           False              False             True
1           False              False            False
2           False              False            False
3           False               True            False
4           False              False            False
5            True              False            False
```

Final Summary of 7 Core Techniques:

1. Variable Transformation: Cleaned symbols, converted types
2. Missing Value Treatment: Used mean and mode
3. Univariate Analysis: Distplots, histograms
4. Outlier Detection: Visually via plots
5. Bivariate Analysis: Regression plots with seaborn
6. Variable Identification: Selected X_iv and y_dv
7. Variable Creation: One-hot encoded categorical variables