1)Replace the NaN values with correct value. And justify why you have chosen the same.

In [1]:
```
import pandas as pd
```

In [2]:
```
dataset=pd.read_csv("Placement_Data_Full_Class.csv")
```

In [3]:
```
dataset
```

Out[3]:

	sl_no	gender	ssc_p	ssc_b	hsc_p	hsc_b	hsc_s	degree_p	degree_t	workex	etest_p	specialisation	mba_p	status	salary
0	1.0	M	67.00	Others	91.00	Others	Commerce	58.00	Sci&Tech	No	55.0	Mkt&HR	58.80	Placed	270000.0
1	2.0	M	79.33	Central	78.33	Others	Science	77.48	Sci&Tech	Yes	86.5	Mkt&Fin	66.28	Placed	200000.0
2	3.0	M	65.00	Central	68.00	Central	Arts	64.00	Comm&Mgmt	No	75.0	Mkt&Fin	57.80	Placed	250000.0
3	4.0	M	56.00	Central	52.00	Central	Science	52.00	Sci&Tech	No	66.0	Mkt&HR	59.43	Not Placed	NaN
4	5.0	M	85.80	Central	73.60	Central	Commerce	73.30	Comm&Mgmt	No	96.8	Mkt&Fin	55.50	Placed	425000.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
212	211.0	M	80.60	Others	82.00	Others	Commerce	77.60	Comm&Mgmt	No	91.0	Mkt&Fin	74.49	Placed	400000.0
213	212.0	M	58.00	Others	60.00	Others	Science	72.00	Sci&Tech	No	74.0	Mkt&Fin	53.62	Placed	275000.0
214	213.0	M	67.00	Others	67.00	Others	Commerce	73.00	Comm&Mgmt	Yes	59.0	Mkt&Fin	69.72	Placed	295000.0
215	214.0	F	74.00	Others	66.00	Others	Commerce	58.00	Comm&Mgmt	No	70.0	Mkt&HR	60.23	Placed	204000.0
216	215.0	M	62.00	Central	58.00	Others	Science	53.00	Comm&Mgmt	No	89.0	Mkt&HR	60.22	Not Placed	NaN

217 rows × 15 columns

In [4]:
```
dataset.isnull().sum()
```

Out[4]:
```
sl_no              2
gender             2
ssc_p              2
ssc_b              2
hsc_p              2
hsc_b              2
hsc_s              2
degree_p           2
degree_t           2
workex             2
etest_p            2
specialisation     2
mba_p              2
status             2
salary            69
dtype: int64
```

In [5]:
```
dataset.describe()
```

Out[5]:

	sl_no	ssc_p	hsc_p	degree_p	etest_p	mba_p	salary
count	215.000000	215.000000	215.000000	215.000000	215.000000	215.000000	148.000000
mean	108.000000	67.303395	66.333163	66.370186	72.100558	62.278186	288655.405405
std	62.209324	10.827205	10.897509	7.358743	13.275956	5.833385	93457.452420
min	1.000000	40.890000	37.000000	50.000000	50.000000	51.210000	200000.000000
25%	54.500000	60.600000	60.900000	61.000000	60.000000	57.945000	240000.000000
50%	108.000000	67.000000	65.000000	66.000000	71.000000	62.000000	265000.000000
75%	161.500000	75.700000	73.000000	72.000000	83.500000	66.255000	300000.000000
max	215.000000	89.400000	97.700000	91.000000	98.000000	77.890000	940000.000000

In [6]:
```
dataset["salary"].fillna(0,inplace=True) #using this updating salary column place 0(zero),
#"inplace=True" means for this dataset we have not assigned any variables. if we assigned varaiable means "inplace=True" not need
```

In [7]:
```
dataset.isnull().sum()
```

Out[7]:
```
sl_no             2
gender            2
ssc_p             2
ssc_b             2
hsc_p             2
hsc_b             2
hsc_s             2
degree_p          2
degree_t          2
workex            2
etest_p           2
specialisation    2
mba_p             2
status            2
salary            0
dtype: int64
```

In [8]:
```
dataset.isna().describe()
```

Out[8]:

	sl_no	gender	ssc_p	ssc_b	hsc_p	hsc_b	hsc_s	degree_p	degree_t	workex	etest_p	specialisation	mba_p	status	salary
count	217	217	217	217	217	217	217	217	217	217	217	217	217	217	217
unique	2	2	2	2	2	2	2	2	2	2	2	2	2	2	1
top	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False
freq	215	215	215	215	215	215	215	215	215	215	215	215	215	215	217

In [10]:
```
# Display rows with any null values
df = pd.DataFrame(dataset)
rows_with_nulls = df[df.isnull().any(axis=1)]
print(rows_with_nulls)
```

```
   sl_no gender  ssc_p ssc_b  hsc_p hsc_b hsc_s  degree_p degree_t workex  \
7    NaN    NaN    NaN   NaN    NaN   NaN   NaN       NaN      NaN    NaN
8    NaN    NaN    NaN   NaN    NaN   NaN   NaN       NaN      NaN    NaN

   etest_p specialisation  mba_p status  salary
7      NaN            NaN    NaN    NaN     0.0
8      NaN            NaN    NaN    NaN     0.0
```

In [11]:
```
# Display rows with null values in column 'sl_no'
rows_with_nulls_in_A = df[df['sl_no'].isnull()]
rows_with_nulls_in_A = df[df['gender'].isnull()]
print(rows_with_nulls_in_A)
```

```
   sl_no gender  ssc_p ssc_b  hsc_p hsc_b hsc_s  degree_p degree_t workex  \
7    NaN    NaN    NaN   NaN    NaN   NaN   NaN       NaN      NaN    NaN
8    NaN    NaN    NaN   NaN    NaN   NaN   NaN       NaN      NaN    NaN

   etest_p specialisation  mba_p status  salary
7      NaN            NaN    NaN    NaN     0.0
8      NaN            NaN    NaN    NaN     0.0
```

In [12]:
```
# Display rows with null values in columns 'A', 'B', and 'C'
rows_with_nulls_in_all = df[df[['sl_no', 'gender', 'ssc_p','ssc_b', 'hsc_p', 'hsc_b','hsc_s', 'degree_p', 'degree_t','workex', 'e
print(rows_with_nulls_in_all)
```

```
Empty DataFrame
Columns: [sl_no, gender, ssc_p, ssc_b, hsc_p, hsc_b, hsc_s, degree_p, degree_t, workex, etest_p, specialisation, mba_p, status,
salary]
Index: []
```

In [13]:
```
# Now decided to delete, Entire row delete
#now we see how to delete/drop entire row
df_dropped_rows = dataset.dropna(inplace=True) #For all Nan value cells/rows deleting into this dataset
print(df_dropped_rows)
```

```
None
```

In [14]:
```
dataset.isna().sum() #isna or isnull both will check Null values only.
```

Out[14]:
```
sl_no             0
gender            0
ssc_p             0
ssc_b             0
hsc_p             0
hsc_b             0
hsc_s             0
degree_p          0
degree_t          0
workex            0
etest_p           0
specialisation    0
mba_p             0
status            0
salary            0
dtype: int64
```

In [15]:
```
dataset
```

Out[15]:

	sl_no	gender	ssc_p	ssc_b	hsc_p	hsc_b	hsc_s	degree_p	degree_t	workex	etest_p	specialisation	mba_p	status	salary
0	1.0	M	67.00	Others	91.00	Others	Commerce	58.00	Sci&Tech	No	55.0	Mkt&HR	58.80	Placed	270000.0
1	2.0	M	79.33	Central	78.33	Others	Science	77.48	Sci&Tech	Yes	86.5	Mkt&Fin	66.28	Placed	200000.0
2	3.0	M	65.00	Central	68.00	Central	Arts	64.00	Comm&Mgmt	No	75.0	Mkt&Fin	57.80	Placed	250000.0
3	4.0	M	56.00	Central	52.00	Central	Science	52.00	Sci&Tech	No	66.0	Mkt&HR	59.43	Not Placed	0.0
4	5.0	M	85.80	Central	73.60	Central	Commerce	73.30	Comm&Mgmt	No	96.8	Mkt&Fin	55.50	Placed	425000.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
212	211.0	M	80.60	Others	82.00	Others	Commerce	77.60	Comm&Mgmt	No	91.0	Mkt&Fin	74.49	Placed	400000.0
213	212.0	M	58.00	Others	60.00	Others	Science	72.00	Sci&Tech	No	74.0	Mkt&Fin	53.62	Placed	275000.0
214	213.0	M	67.00	Others	67.00	Others	Commerce	73.00	Comm&Mgmt	Yes	59.0	Mkt&Fin	69.72	Placed	295000.0
215	214.0	F	74.00	Others	66.00	Others	Commerce	58.00	Comm&Mgmt	No	70.0	Mkt&HR	60.23	Placed	204000.0
216	215.0	M	62.00	Central	58.00	Others	Science	53.00	Comm&Mgmt	No	89.0	Mkt&HR	60.22	Not Placed	0.0

215 rows × 15 columns

In [17]:
```
dataset.isnull().sum() #checking any null dataset available or not
```

Out[17]:
```
sl_no             0
gender            0
ssc_p             0
ssc_b             0
hsc_p             0
hsc_b             0
hsc_s             0
degree_p          0
degree_t          0
workex            0
etest_p           0
specialisation    0
mba_p             0
status            0
salary            0
dtype: int64
```

In [19]:
```
dataset.to_csv("Placement_Data_Full_Class_Preprocessed.csv",index=False) #inde=false means it'll not create duplicate index, now
```

2)How many of them are not placed?

In [20]:
```
dataset=pd.read_csv("Placement_Data_Full_Class.csv")
```

In [21]:
```
dataset
```

Out[21]:

	sl_no	gender	ssc_p	ssc_b	hsc_p	hsc_b	hsc_s	degree_p	degree_t	workex	etest_p	specialisation	mba_p	status	salary
0	1.0	M	67.00	Others	91.00	Others	Commerce	58.00	Sci&Tech	No	55.0	Mkt&HR	58.80	Placed	270000.0
1	2.0	M	79.33	Central	78.33	Others	Science	77.48	Sci&Tech	Yes	86.5	Mkt&Fin	66.28	Placed	200000.0
2	3.0	M	65.00	Central	68.00	Central	Arts	64.00	Comm&Mgmt	No	75.0	Mkt&Fin	57.80	Placed	250000.0
3	4.0	M	56.00	Central	52.00	Central	Science	52.00	Sci&Tech	No	66.0	Mkt&HR	59.43	Not Placed	NaN
4	5.0	M	85.80	Central	73.60	Central	Commerce	73.30	Comm&Mgmt	No	96.8	Mkt&Fin	55.50	Placed	425000.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
212	211.0	M	80.60	Others	82.00	Others	Commerce	77.60	Comm&Mgmt	No	91.0	Mkt&Fin	74.49	Placed	400000.0
213	212.0	M	58.00	Others	60.00	Others	Science	72.00	Sci&Tech	No	74.0	Mkt&Fin	53.62	Placed	275000.0
214	213.0	M	67.00	Others	67.00	Others	Commerce	73.00	Comm&Mgmt	Yes	59.0	Mkt&Fin	69.72	Placed	295000.0
215	214.0	F	74.00	Others	66.00	Others	Commerce	58.00	Comm&Mgmt	No	70.0	Mkt&HR	60.23	Placed	204000.0
216	215.0	M	62.00	Central	58.00	Others	Science	53.00	Comm&Mgmt	No	89.0	Mkt&HR	60.22	Not Placed	NaN

217 rows × 15 columns

In [22]:
```
#We want only status column details of 'Not Placed' student details, it'll retrive only 'Not Placed' student data using python
dataset[dataset['status']=='Not Placed']
```

Out[22]:

	sl_no	gender	ssc_p	ssc_b	hsc_p	hsc_b	hsc_s	degree_p	degree_t	workex	etest_p	specialisation	mba_p	status	salary
3	4.0	M	56.0	Central	52.0	Central	Science	52.00	Sci&Tech	No	66.00	Mkt&HR	59.43	Not Placed	NaN
5	6.0	M	55.0	Others	49.8	Others	Science	67.25	Sci&Tech	Yes	55.00	Mkt&Fin	51.58	Not Placed	NaN
6	7.0	F	46.0	Others	49.2	Others	Commerce	79.00	Comm&Mgmt	No	74.28	Mkt&Fin	53.29	Not Placed	NaN
11	10.0	M	58.0	Central	70.0	Central	Commerce	61.00	Comm&Mgmt	No	54.00	Mkt&Fin	52.21	Not Placed	NaN
14	13.0	F	47.0	Central	55.0	Others	Science	65.00	Comm&Mgmt	No	62.00	Mkt&HR	65.04	Not Placed	NaN
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
200	199.0	F	67.0	Central	70.0	Central	Commerce	65.00	Others	No	88.00	Mkt&HR	71.96	Not Placed	NaN
203	202.0	M	54.2	Central	63.0	Others	Science	58.00	Comm&Mgmt	No	79.00	Mkt&HR	58.44	Not Placed	NaN
208	207.0	M	41.0	Central	42.0	Central	Science	60.00	Comm&Mgmt	No	97.00	Mkt&Fin	53.39	Not Placed	NaN
210	209.0	F	43.0	Central	60.0	Others	Science	65.00	Comm&Mgmt	No	92.66	Mkt&HR	62.92	Not Placed	NaN
216	215.0	M	62.0	Central	58.0	Others	Science	53.00	Comm&Mgmt	No	89.00	Mkt&HR	60.22	Not Placed	NaN

67 rows × 15 columns

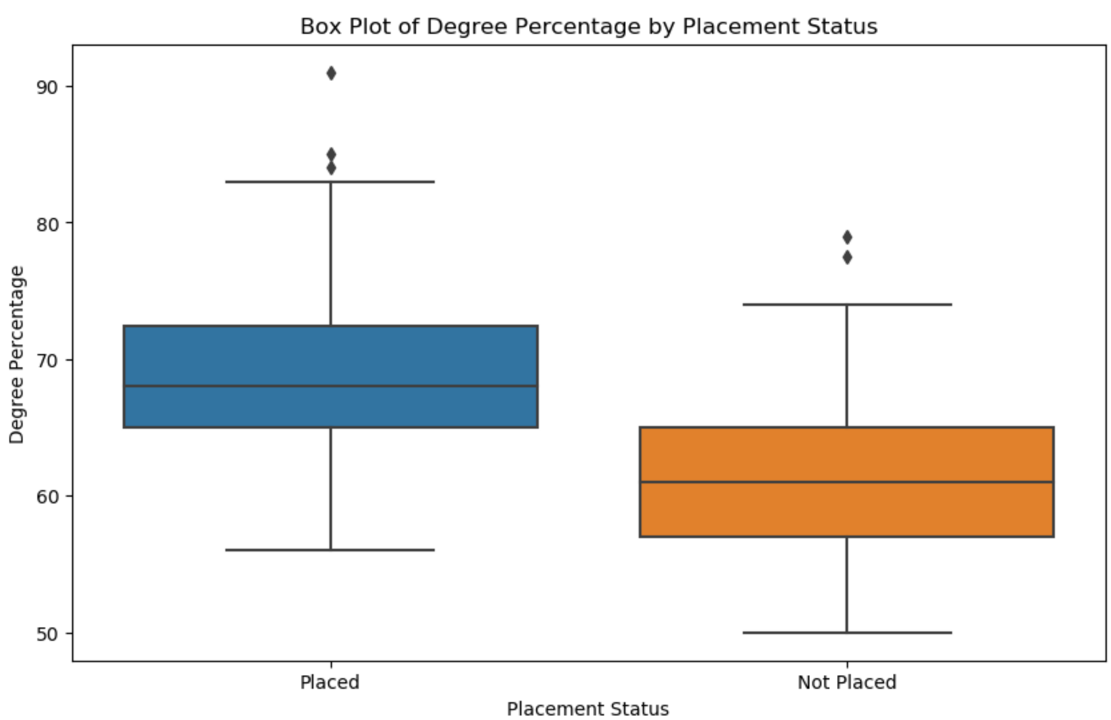
3)Find the reason for non-placement from the dataset?

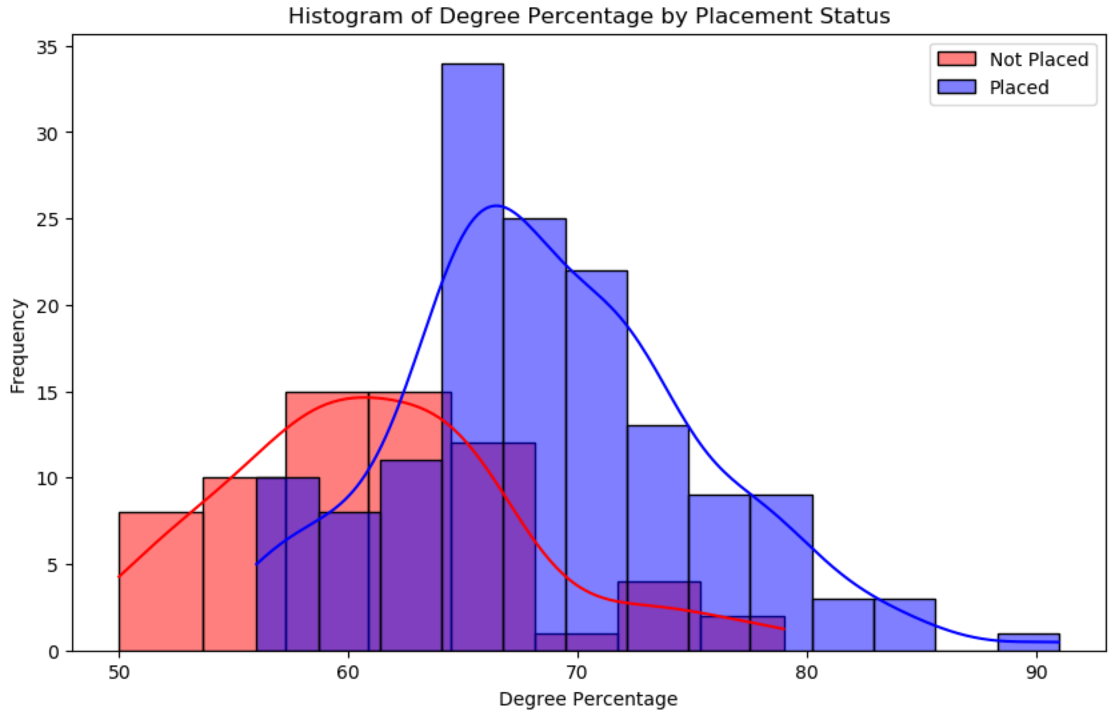
In [23]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Load the dataset
df = pd.read_csv('Placement_Data_Full_Class.csv')

# Display the first few rows to inspect the structure
print(df.head())

# Filter data for non-placed students
non_placed = df[df['status'] == 'Not Placed']
placed = df[df['status'] == 'Placed']

# Summary statistics for numerical columns
print("Summary statistics for non-placed students:")
print(non_placed.describe())

print("\nSummary statistics for placed students:")
print(placed.describe())

# Visualizing differences between placed and non-placed students

# Plot for numerical attributes
numerical_columns = ['ssc_p', 'hsc_p', 'degree_p', 'etest_p', 'mba_p']

plt.figure(figsize=(15, 10))
for i, col in enumerate(numerical_columns, 1):
    plt.subplot(2, 3, i)
    sns.histplot(data=df, x=col, hue='status', kde=True, element="step")
    plt.title(f'Distribution of {col}')
plt.tight_layout()
plt.show()

# Plot for categorical attributes
categorical_columns = ['gender', 'workex', 'specialisation']

plt.figure(figsize=(15, 10))
for i, col in enumerate(categorical_columns, 1):
    plt.subplot(2, 3, i)
    sns.countplot(data=df, x=col, hue='status')
    plt.title(f'Count of {col}')
plt.tight_layout()
plt.show()
```

```
   sl_no gender  ssc_p    ssc_b  hsc_p    hsc_b     hsc_s  degree_p  \
0    1.0      M  67.00   Others  91.00   Others  Commerce     58.00
1    2.0      M  79.33  Central  78.33   Others   Science     77.48
2    3.0      M  65.00  Central  68.00  Central      Arts     64.00
3    4.0      M  56.00  Central  52.00  Central   Science     52.00
4    5.0      M  85.80  Central  73.60  Central  Commerce     73.30

    degree_t workex  etest_p specialisation  mba_p      status    salary
0   Sci&Tech     No     55.0         Mkt&HR  58.80      Placed  270000.0
1   Sci&Tech    Yes     86.5        Mkt&Fin  66.28      Placed  200000.0
2  Comm&Mgmt     No     75.0        Mkt&Fin  57.80      Placed  250000.0
3   Sci&Tech     No     66.0         Mkt&HR  59.43  Not Placed       NaN
4  Comm&Mgmt     No     96.8        Mkt&Fin  55.50      Placed  425000.0
Summary statistics for non-placed students:
            sl_no      ssc_p      hsc_p   degree_p    etest_p      mba_p  \
count   67.000000  67.000000  67.000000  67.000000  67.000000  67.000000
mean   110.477612  57.544030  58.395522  61.134179  69.587910  61.612836
std     65.859667   8.394246   9.914090   6.365825  11.930687   5.705689
min      4.000000  40.890000  37.000000  50.000000  50.000000  51.210000
```

In [24]:
```python
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Load the dataset
df = pd.read_csv('Placement_Data_Full_Class.csv')

# Display the first few rows to inspect the structure
print(df.head())

# Filter data for placed and non-placed students
non_placed = df[df['status'] == 'Not Placed']
placed = df[df['status'] == 'Placed']

# Visualizing the distribution of 'degree_p' for placed and non-placed students

# Box plot
plt.figure(figsize=(10, 6))
sns.boxplot(x='status', y='degree_p', data=df)
plt.title('Box Plot of Degree Percentage by Placement Status')
plt.xlabel('Placement Status')
plt.ylabel('Degree Percentage')
plt.show()

# Histogram
plt.figure(figsize=(10, 6))
sns.histplot(non_placed['degree_p'], color='red', label='Not Placed', kde=True)
sns.histplot(placed['degree_p'], color='blue', label='Placed', kde=True)
plt.title('Histogram of Degree Percentage by Placement Status')
plt.xlabel('Degree Percentage')
plt.ylabel('Frequency')
plt.legend()
plt.show()
```

```
   sl_no gender  ssc_p    ssc_b  hsc_p    hsc_b     hsc_s  degree_p  \
0    1.0      M  67.00   Others  91.00   Others  Commerce     58.00
1    2.0      M  79.33  Central  78.33   Others   Science     77.48
2    3.0      M  65.00  Central  68.00  Central      Arts     64.00
3    4.0      M  56.00  Central  52.00  Central   Science     52.00
4    5.0      M  85.80  Central  73.60  Central  Commerce     73.30

    degree_t workex  etest_p specialisation  mba_p      status    salary
0   Sci&Tech     No     55.0         Mkt&HR  58.80      Placed  270000.0
1   Sci&Tech    Yes     86.5        Mkt&Fin  66.28      Placed  200000.0
2  Comm&Mgmt     No     75.0        Mkt&Fin  57.80      Placed  250000.0
3   Sci&Tech     No     66.0         Mkt&HR  59.43  Not Placed       NaN
4  Comm&Mgmt     No     96.8        Mkt&Fin  55.50      Placed  425000.0
```

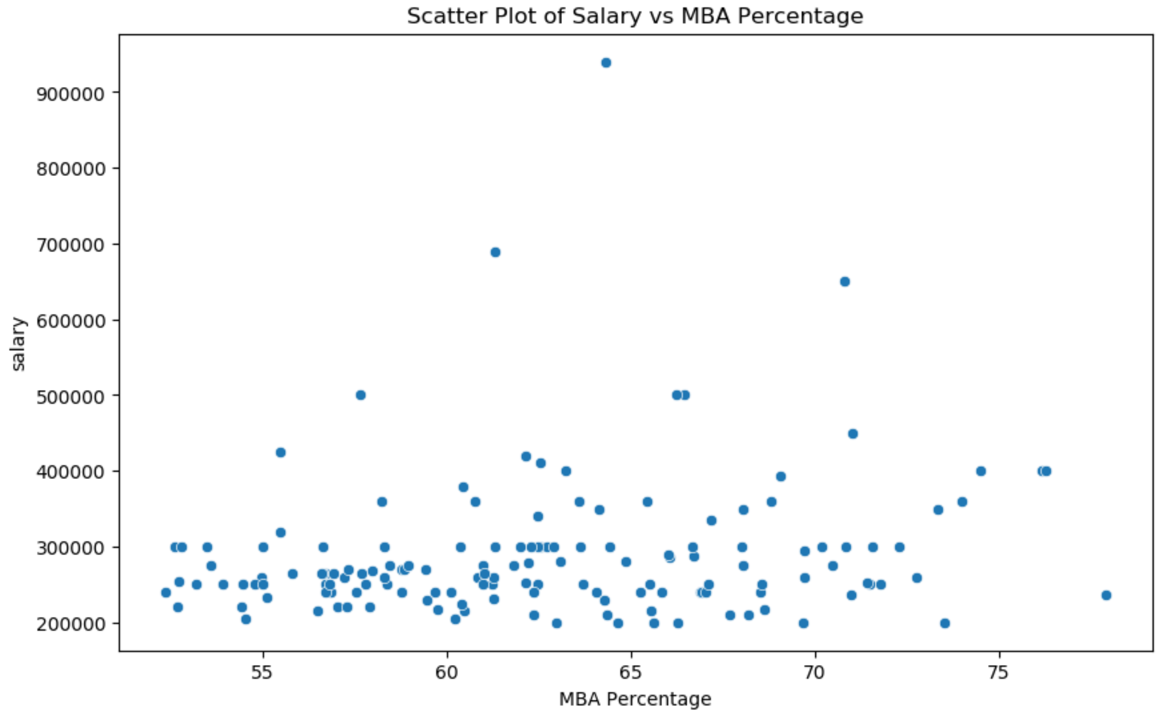




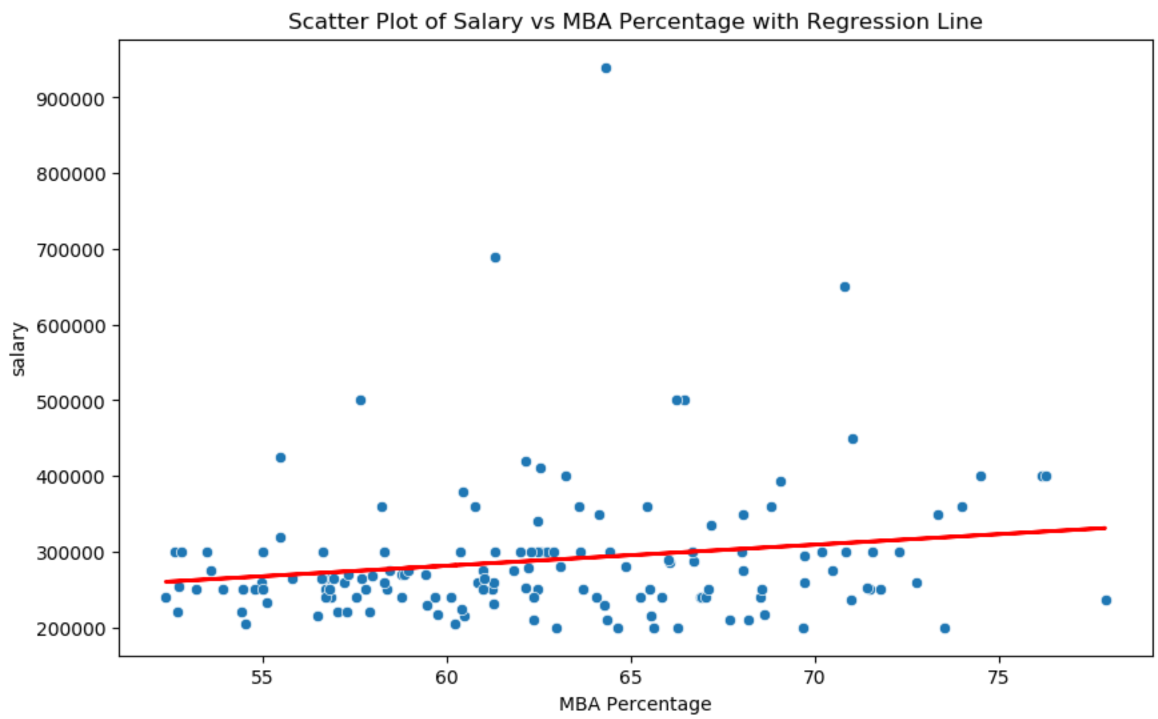
4)What kind of relation between salary and mba_p?

In [26]:
```python
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import pearsonr
from sklearn.linear_model import LinearRegression
import numpy as np

# Load the dataset
df = pd.read_csv('Placement_Data_Full_Class.csv')

# Inspect the data
print(df.head())

# Handle missing values by dropping rows with missing salary or mba_p
df = df.dropna(subset=['salary', 'mba_p'])

# Scatter plot to visualize the relationship
plt.figure(figsize=(10, 6))
sns.scatterplot(data=df, x='mba_p', y='salary')
plt.title('Scatter Plot of Salary vs MBA Percentage')
plt.xlabel('MBA Percentage')
plt.ylabel('salary')
plt.show()

# Calculate the Pearson correlation coefficient
correlation, p_value = pearsonr(df['mba_p'], df['salary'])
print(f'Pearson correlation coefficient: {correlation}')
print(f'P-value: {p_value}')

# Fit a simple linear regression model
X = df[['mba_p']]
y = df['salary']

linear_regressor = LinearRegression()
linear_regressor.fit(X, y)

# Predict salary based on the MBA percentage
y_pred = linear_regressor.predict(X)

# Plot the regression line
plt.figure(figsize=(10, 6))
sns.scatterplot(data=df, x='mba_p', y='salary')
plt.plot(df['mba_p'], y_pred, color='red', linewidth=2)
plt.title('Scatter Plot of Salary vs MBA Percentage with Regression Line')
plt.xlabel('MBA Percentage')
plt.ylabel('salary')
plt.show()

# Display the regression equation
slope = linear_regressor.coef_[0]
intercept = linear_regressor.intercept_
print(f'Regression equation: salary = {intercept:.2f} + {slope:.2f} * MBA Percentage')
```

```
   sl_no gender  ssc_p    ssc_b  hsc_p    hsc_b     hsc_s  degree_p  \
0    1.0      M  67.00   Others  91.00   Others  Commerce     58.00
1    2.0      M  79.33  Central  78.33   Others   Science     77.48
2    3.0      M  65.00  Central  68.00  Central      Arts     64.00
3    4.0      M  56.00  Central  52.00  Central   Science     52.00
4    5.0      M  85.80  Central  73.60  Central  Commerce     73.30

    degree_t workex  etest_p specialisation  mba_p      status    salary
0   Sci&Tech     No     55.0         Mkt&HR  58.80      Placed  270000.0
1   Sci&Tech    Yes     86.5        Mkt&Fin  66.28      Placed  200000.0
2  Comm&Mgmt     No     75.0        Mkt&Fin  57.80      Placed  250000.0
3   Sci&Tech     No     66.0         Mkt&HR  59.43  Not Placed       NaN
4  Comm&Mgmt     No     96.8        Mkt&Fin  55.50      Placed  425000.0
```




```
Pearson correlation coefficient: 0.17501294069527484
P-value: 0.03337689255770916
```




```
Regression equation: salary = 114715.29 + 2779.51 * MBA Percentage
```

In [27]:
```
dataset.corr()
#Always correlation cross value or linear(1.000000) will be same.correlation diagonal value will be same(1.000000).
#correlation value will be same or repeated for diagonal upper and lower side.correlation value(1) means exact match.
#Two columns relationship check using correlation. but covaraiance we are using to differnce between two columns.
 #the correlation between salary and mba_p - 0.139823 (Positive correlation)
```

Out[27]:

	sl_no	ssc_p	hsc_p	degree_p	etest_p	mba_p	salary
sl_no	1.000000	-0.078155	-0.085711	-0.088281	0.063636	0.022327	0.063764
ssc_p	-0.078155	1.000000	0.511472	0.538404	0.261993	0.388478	0.035330
hsc_p	-0.085711	0.511472	1.000000	0.434206	0.245113	0.354823	0.076819
degree_p	-0.088281	0.538404	0.434206	1.000000	0.224470	0.402364	-0.019272
etest_p	0.063636	0.261993	0.245113	0.224470	1.000000	0.218055	0.178307
mba_p	0.022327	0.388478	0.354823	0.402364	0.218055	1.000000	0.175013
salary	0.063764	0.035330	0.076819	-0.019272	0.178307	0.175013	1.000000

5)Which specialization is getting minimum salary?

In [29]:
```
import pandas as pd

# Load the dataset
df = pd.read_csv('Placement_Data_Full_Class.csv')

# Display the first few rows to inspect the structure
print(df.head())

# Handle missing values in the Salary column (choose one method)
df['salary'].fillna(0, inplace=True)  # Fill missing salary values with 0
# df.dropna(subset=['Salary'], inplace=True)  # Or drop rows with missing salary values

# Ensure the relevant columns are present
if 'specialisation' in df.columns and 'salary' in df.columns:
    # Group by Specialization and calculate the minimum salary
    min_salary_by_specialization = df.groupby('specialisation')['salary'].min()

    # Display the result
    print(min_salary_by_specialization)

    # Find the specialization with the minimum salary
    min_salary_specialization = min_salary_by_specialization.idxmin()
    min_salary_value = min_salary_by_specialization.min()

    print(f"The specialization with the minimum salary is {min_salary_specialization} with a salary of {min_salary_value}.")
else:
    print("The required columns are not present in the dataset.")
```

```
   sl_no gender  ssc_p    ssc_b  hsc_p    hsc_b     hsc_s  degree_p  \
0    1.0      M  67.00   Others  91.00   Others  Commerce     58.00   
1    2.0      M  79.33  Central  78.33   Others   Science     77.48   
2    3.0      M  65.00  Central  68.00  Central      Arts     64.00   
3    4.0      M  56.00  Central  52.00  Central   Science     52.00   
4    5.0      M  85.80  Central  73.60  Central  Commerce     73.30   

    degree_t workex  etest_p specialisation  mba_p      status    salary  
0   Sci&Tech     No     55.0         Mkt&HR  58.80      Placed  270000.0  
1   Sci&Tech    Yes     86.5        Mkt&Fin  66.28      Placed  200000.0  
2  Comm&Mgmt     No     75.0        Mkt&Fin  57.80      Placed  250000.0  
3   Sci&Tech     No     66.0         Mkt&HR  59.43  Not Placed       NaN  
4  Comm&Mgmt     No     96.8        Mkt&Fin  55.50      Placed  425000.0  
specialisation
Mkt&Fin    0.0
Mkt&HR     0.0
Name: salary, dtype: float64
The specialization with the minimum salary is Mkt&Fin with a salary of 0.0.
```

In [30]:
```
df.groupby('specialisation')['salary'].min()
```

Out[30]:
```
specialisation
Mkt&Fin    0.0
Mkt&HR     0.0
Name: salary, dtype: float64
```

6)How many of them getting above 500000 salaries?

In [31]:
```
Highsalary=dataset[dataset['salary'] > 500000 ]
```

In [32]:
```
Highsalary
```

Out[32]:

	sl_no	gender	ssc_p	ssc_b	hsc_p	hsc_b	hsc_s	degree_p	degree_t	workex	etest_p	specialisation	mba_p	status	salary
121	120.0	M	60.8	Central	68.40	Central	Commerce	64.6	Comm&Mgmt	Yes	82.66	Mkt&Fin	64.34	Placed	940000.0
152	151.0	M	71.0	Central	58.66	Central	Science	58.0	Sci&Tech	Yes	56.00	Mkt&Fin	61.30	Placed	690000.0
179	178.0	F	73.0	Central	97.00	Others	Commerce	79.0	Comm&Mgmt	Yes	89.00	Mkt&Fin	70.81	Placed	650000.0

In [33]:
```
import pandas as pd

# Load the dataset
df = pd.read_csv('Placement_Data_Full_Class.csv')

# Display the first few rows to inspect the structure
print(df.head())

# Handle missing values in the Salary column
df.dropna(subset=['salary'], inplace=True)  # Drop rows with missing salary values

# Ensure the 'Salary' column is present
if 'salary' in df.columns:
    # Filter the DataFrame for salaries above 500,000
    high_salary_df = df[df['salary'] > 500000]

    # Count the number of entries with salary above 500,000
    count_high_salary = high_salary_df.shape[0]

    print(f"Number of students with salaries above 500,000: {count_high_salary}")
else:
    print("The 'Salary' column is not present in the dataset.")
```

```
   sl_no gender  ssc_p    ssc_b  hsc_p    hsc_b     hsc_s  degree_p  \
0    1.0      M  67.00   Others  91.00   Others  Commerce     58.00
1    2.0      M  79.33  Central  78.33   Others   Science     77.48
2    3.0      M  65.00  Central  68.00  Central      Arts     64.00
3    4.0      M  56.00  Central  52.00  Central   Science     52.00
4    5.0      M  85.80  Central  73.60  Central  Commerce     73.30

    degree_t workex  etest_p specialisation  mba_p      status    salary
0   Sci&Tech     No     55.0         Mkt&HR  58.80      Placed  270000.0
1   Sci&Tech    Yes     86.5        Mkt&Fin  66.28      Placed  200000.0
2  Comm&Mgmt     No     75.0        Mkt&Fin  57.80      Placed  250000.0
3   Sci&Tech     No     66.0         Mkt&HR  59.43  Not Placed       NaN
4  Comm&Mgmt     No     96.8        Mkt&Fin  55.50      Placed  425000.0
Number of students with salaries above 500,000: 3
```

7)Test the Analysis of Variance between etest_p and mba_p at signifance

level 5%.(Make decision using Hypothesis Testing)

In [43]:
```
import pandas as pd
from scipy.stats import f_oneway

# Load the dataset
df = pd.read_csv('Placement_Data_Full_Class.csv')

# Drop rows with missing values in 'etest_p' or 'mba_p'
df = df.dropna(subset=['etest_p', 'mba_p'])

# Perform ANOVA(Analysis of Variance)
f_stat, p_value = f_oneway(df['etest_p'], df['mba_p'])

# Output the results
print(f'F-statistic: {f_stat}')
print(f'P-value: {p_value}')

# Significance Level
alpha = 0.05

# Decision based on the p-value
if p_value < alpha:
    print("Reject the null hypothesis. There is a significant difference between the means of etest_p and mba_p.")
else:
    print("Fail to reject the null hypothesis. There is no significant difference between the means of etest_p and mba_p.")
```

```
F-statistic: 98.64487057324706
P-value: 4.672547689133573e-21
Reject the null hypothesis. There is a significant difference between the means of etest_p and mba_p.
```

In [35]:
```
dataset.cov()
```

Out[35]:

	sl_no	ssc_p	hsc_p	degree_p	etest_p	mba_p	salary
sl_no	3870.000000	-52.641355	-58.106028	-40.413645	52.556168	8.102336	3.616177e+05
ssc_p	-52.641355	117.228377	60.348373	42.897137	37.659225	24.535952	2.877739e+04
hsc_p	-58.106028	60.348373	118.755706	34.819820	35.461678	22.555846	6.697772e+04
degree_p	-40.413645	42.897137	34.819820	54.151103	21.929469	17.272020	-1.173995e+04
etest_p	52.556168	37.659225	35.461678	21.929469	176.251018	16.886973	2.287876e+05
mba_p	8.102336	24.535952	22.555846	17.272020	16.886973	34.028376	9.624979e+04
salary	361617.668689	28777.386468	66977.716032	-11739.948520	228787.619507	96249.789024	8.734295e+09

In [36]:
```
dataset.corr()
```

Out[36]:

	sl_no	ssc_p	hsc_p	degree_p	etest_p	mba_p	salary
sl_no	1.000000	-0.078155	-0.085711	-0.088281	0.063636	0.022327	0.063764
ssc_p	-0.078155	1.000000	0.511472	0.538404	0.261993	0.388478	0.035330
hsc_p	-0.085711	0.511472	1.000000	0.434206	0.245113	0.354823	0.076819
degree_p	-0.088281	0.538404	0.434206	1.000000	0.224470	0.402364	-0.019272
etest_p	0.063636	0.261993	0.245113	0.224470	1.000000	0.218055	0.178307
mba_p	0.022327	0.388478	0.354823	0.402364	0.218055	1.000000	0.175013
salary	0.063764	0.035330	0.076819	-0.019272	0.178307	0.175013	1.000000

In [37]:
```
#7)Test the Analysis of Variance between etest_p and mba_p at signifance level 5%.(Make decision using Hypothesis Testing)
#Found Commerce students MBA pass mark and science group studied peoples Mba passmark difference.
from scipy.stats import ttest_rel
#dataset=dataset.dropna()
male = dataset[dataset['gender']=='M']['etest_p']
male1 = dataset[dataset['gender']=='M']['mba_p']
#print(male)
ttest_rel(male,male1)
```

Out[37]:
```
Ttest_relResult(statistic=10.558377508518305, pvalue=1.8140319084982095e-19)
```

8)Test the similarity between the degree_t(Sci&Tech) and specialisation (Mkt&HR) with respect to salary at significance level of 5%.(Make decision using Hypothesis Testing)

In [44]:
```
import pandas as pd
from scipy.stats import ttest_ind

# Load the dataset
df = pd.read_csv('Placement_Data_Full_Class.csv')

# Filter salaries for 'Sci&Tech' degree_t
sci_tech_salaries = df[df['degree_t'] == 'Sci&Tech']['salary'].dropna()

# Filter salaries for 'Mkt&HR' specialisation
mkt_hr_salaries = df[df['specialisation'] == 'Mkt&HR']['salary'].dropna()

# Perform the t-test
t_stat, p_value = ttest_ind(sci_tech_salaries, mkt_hr_salaries)

# Output the results
print(f'T-statistic: {t_stat}')
print(f'P-value: {p_value}')

# Significance level
alpha = 0.05

# Decision based on the p-value
if p_value < alpha:
    print("Reject the null hypothesis. There is a significant difference in salaries.")
else:
    print("Fail to reject the null hypothesis. There is no significant difference in salaries.")
```

```
T-statistic: 2.734391160944239
P-value: 0.007496896218767113
Reject the null hypothesis. There is a significant difference in salaries.
```

9)Convert the normal distribution to standard normal distribution for salary column

In [46]:
```
import pandas as pd

# Load the dataset
df = pd.read_csv('Placement_Data_Full_Class.csv')

# Calculate mean and standard deviation of the Salary column
salary_mean = df['salary'].mean()
salary_std = df['salary'].std()

print(f"Mean of Salary: {salary_mean}")
print(f"Standard Deviation of Salary: {salary_std}")

# Standardize the Salary column
df['Salary_standardized'] = (df['salary'] - salary_mean) / salary_std

# Display the first few rows to check the new column
print(df[['salary', 'Salary_standardized']].head())
```

```
Mean of Salary: 288655.4054054054
Standard Deviation of Salary: 93457.45241958876
     salary  Salary_standardized
0  270000.0            -0.199614
1  200000.0            -0.948618
2  250000.0            -0.413615
3       NaN                  NaN
4  425000.0             1.458895
```

10)What is the probability Density Function of the salary range from 700000 to 900000?

In [48]:
```
import pandas as pd
import numpy as np
from scipy.stats import norm

# Load the dataset
df = pd.read_csv('Placement_Data_Full_Class.csv')

# Calculate mean and standard deviation of the Salary column
salary_mean = df['salary'].mean()
salary_std = df['salary'].std()

print(f"Mean of salary: {salary_mean}")
print(f"Standard Deviation of salary: {salary_std}")

# Define the range of salaries
salary_range = np.linspace(700000, 900000, 100)

# Calculate the PDF values for this range
pdf_values = norm.pdf(salary_range, salary_mean, salary_std)

# Display the range and corresponding PDF values
for salary, pdf in zip(salary_range, pdf_values):
    print(f"salary: {salary:.2f}, PDF: {pdf:.6f}")
```

```
Mean of salary: 288655.4054054054
Standard Deviation of salary: 93457.45241958876
salary: 700000.00, PDF: 0.000000
salary: 702020.20, PDF: 0.000000
salary: 704040.40, PDF: 0.000000
salary: 706060.61, PDF: 0.000000
salary: 708080.81, PDF: 0.000000
salary: 710101.01, PDF: 0.000000
salary: 712121.21, PDF: 0.000000
salary: 714141.41, PDF: 0.000000
salary: 716161.62, PDF: 0.000000
salary: 718181.82, PDF: 0.000000
salary: 720202.02, PDF: 0.000000
salary: 722222.22, PDF: 0.000000
salary: 724242.42, PDF: 0.000000
salary: 726262.63, PDF: 0.000000
salary: 728282.83, PDF: 0.000000
salary: 730303.03, PDF: 0.000000
salary: 732323.23, PDF: 0.000000
salary: 734343.43, PDF: 0.000000
salary: 736363.64, PDF: 0.000000
salary: 738383.84, PDF: 0.000000
salary: 740404.04, PDF: 0.000000
salary: 742424.24, PDF: 0.000000
salary: 744444.44, PDF: 0.000000
salary: 746464.65, PDF: 0.000000
salary: 748484.85, PDF: 0.000000
salary: 750505.05, PDF: 0.000000
salary: 752525.25, PDF: 0.000000
salary: 754545.45, PDF: 0.000000
salary: 756565.66, PDF: 0.000000
salary: 758585.86, PDF: 0.000000
salary: 760606.06, PDF: 0.000000
salary: 762626.26, PDF: 0.000000
salary: 764646.46, PDF: 0.000000
salary: 766666.67, PDF: 0.000000
salary: 768686.87, PDF: 0.000000
salary: 770707.07, PDF: 0.000000
salary: 772727.27, PDF: 0.000000
salary: 774747.47, PDF: 0.000000
salary: 776767.68, PDF: 0.000000
salary: 778787.88, PDF: 0.000000
salary: 780808.08, PDF: 0.000000
salary: 782828.28, PDF: 0.000000
salary: 784848.48, PDF: 0.000000
salary: 786868.69, PDF: 0.000000
salary: 788888.89, PDF: 0.000000
salary: 790909.09, PDF: 0.000000
salary: 792929.29, PDF: 0.000000
salary: 794949.49, PDF: 0.000000
salary: 796969.70, PDF: 0.000000
salary: 798989.90, PDF: 0.000000
salary: 801010.10, PDF: 0.000000
salary: 803030.30, PDF: 0.000000
salary: 805050.51, PDF: 0.000000
salary: 807070.71, PDF: 0.000000
salary: 809090.91, PDF: 0.000000
salary: 811111.11, PDF: 0.000000
salary: 813131.31, PDF: 0.000000
salary: 815151.52, PDF: 0.000000
salary: 817171.72, PDF: 0.000000
salary: 819191.92, PDF: 0.000000
salary: 821212.12, PDF: 0.000000
salary: 823232.32, PDF: 0.000000
salary: 825252.53, PDF: 0.000000
salary: 827272.73, PDF: 0.000000
salary: 829292.93, PDF: 0.000000
salary: 831313.13, PDF: 0.000000
salary: 833333.33, PDF: 0.000000
salary: 835353.54, PDF: 0.000000
salary: 837373.74, PDF: 0.000000
salary: 839393.94, PDF: 0.000000
salary: 841414.14, PDF: 0.000000
salary: 843434.34, PDF: 0.000000
salary: 845454.55, PDF: 0.000000
salary: 847474.75, PDF: 0.000000
salary: 849494.95, PDF: 0.000000
salary: 851515.15, PDF: 0.000000
salary: 853535.35, PDF: 0.000000
salary: 855555.56, PDF: 0.000000
salary: 857575.76, PDF: 0.000000
salary: 859595.96, PDF: 0.000000
salary: 861616.16, PDF: 0.000000
salary: 863636.36, PDF: 0.000000
salary: 865656.57, PDF: 0.000000
salary: 867676.77, PDF: 0.000000
```

```
salary: 869696.97, PDF: 0.000000
salary: 871717.17, PDF: 0.000000
salary: 873737.37, PDF: 0.000000
salary: 875757.58, PDF: 0.000000
salary: 877777.78, PDF: 0.000000
salary: 879797.98, PDF: 0.000000
salary: 881818.18, PDF: 0.000000
salary: 883838.38, PDF: 0.000000
salary: 885858.59, PDF: 0.000000
salary: 887878.79, PDF: 0.000000
salary: 889898.99, PDF: 0.000000
salary: 891919.19, PDF: 0.000000
salary: 893939.39, PDF: 0.000000
salary: 895959.60, PDF: 0.000000
salary: 897979.80, PDF: 0.000000
salary: 900000.00, PDF: 0.000000
```

11)Test the similarity between the degree_t(Sci&Tech)with respect to etest_p and mba_p at significance level of 5%.(Make decision using Hypothesis Testing)

In [49]:

```
import pandas as pd
from scipy.stats import ttest_rel

# Load the dataset
df = pd.read_csv('Placement_Data_Full_Class.csv')

# Filter data for 'Sci&Tech' degree_t
sci_tech_data = df[df['degree_t'] == 'Sci&Tech']

# Ensure no missing values in 'etest_p' or 'mba_p'
sci_tech_data = sci_tech_data.dropna(subset=['etest_p', 'mba_p'])

# Perform the paired samples t-test
t_stat, p_value = ttest_rel(sci_tech_data['etest_p'], sci_tech_data['mba_p'])

# Output the results
print(f'T-statistic: {t_stat}')
print(f'P-value: {p_value}')

# Significance level
alpha = 0.05

# Decision based on the p-value
if p_value < alpha:
    print("Reject the null hypothesis. There is a significant difference between etest_p and mba_p for Sci&Tech students.")
else:
    print("Fail to reject the null hypothesis. There is no significant difference between etest_p and mba_p for Sci&Tech students
```

```
T-statistic: 5.0049844583693615
P-value: 5.517920600505392e-06
Reject the null hypothesis. There is a significant difference between etest_p and mba_p for Sci&Tech students.
```

12)Which parameter is highly correlated with salary?

In [50]:
```
import pandas as pd

# Load the dataset
df = pd.read_csv('Placement_Data_Full_Class.csv')

# Calculate the correlation matrix
correlation_matrix = df.corr()

# Extract the correlation of Salary with other columns
salary_correlation = correlation_matrix['salary']

# Find the parameter with the highest correlation with Salary (excluding Salary itself)
highest_correlation = salary_correlation.drop('salary').idxmax()
highest_value = salary_correlation[highest_correlation]

print(salary_correlation)

print(f"The parameter most highly correlated with Salary is {highest_correlation} with a correlation of {highest_value:.2f}")
```

```
sl_no       0.063764
ssc_p       0.035330
hsc_p       0.076819
degree_p   -0.019272
etest_p     0.178307
mba_p       0.175013
salary      1.000000
Name: salary, dtype: float64
The parameter most highly correlated with Salary is etest_p with a correlation of 0.18
```

In [51]:
```
#what is the covaraiance between degree_p and etest_p is 22.078774 -Large positive covaraiance
#what is the covaraiance between etest aand mba_p is 16.886973 -Large positive covaraiance
dataset.corr()
#Always correlation cross value or linear(1.000000) will be same.correlation diagonal value will be same(1.000000).
#correlation value will be same or repeated for diagonal upper and lower side.correlation value(1) means exact match.
#what is correlation between ssc_p and hsc_p- 0.513478 (this value like nuteral, no increase and drcrease for both side).-it's ze
#what is correlation between mba_p and ssc_p relationship -0.388478 (it's nearly to positive correlation, but degree of freedom i
#what is the correlation between mba_p and salary - 0.141417 (Zero correlation)
```

Out[51]:

	sl_no	ssc_p	hsc_p	degree_p	etest_p	mba_p	salary
sl_no	1.000000	-0.078155	-0.085711	-0.088281	0.063636	0.022327	0.063764
ssc_p	-0.078155	1.000000	0.511472	0.538404	0.261993	0.388478	0.035330
hsc_p	-0.085711	0.511472	1.000000	0.434206	0.245113	0.354823	0.076819
degree_p	-0.088281	0.538404	0.434206	1.000000	0.224470	0.402364	-0.019272
etest_p	0.063636	0.261993	0.245113	0.224470	1.000000	0.218055	0.178307
mba_p	0.022327	0.388478	0.354823	0.402364	0.218055	1.000000	0.175013
salary	0.063764	0.035330	0.076819	-0.019272	0.178307	0.175013	1.000000

In [52]:
```
dataset.cov() #it was taken all quantitative columns and processed.index and column name as main coulmns(x and y axis col names c
#ssc_p and hsc_p pass mark diff is 58.853253 so it's positive covaraiance. we can see diff in 58.853253.
```

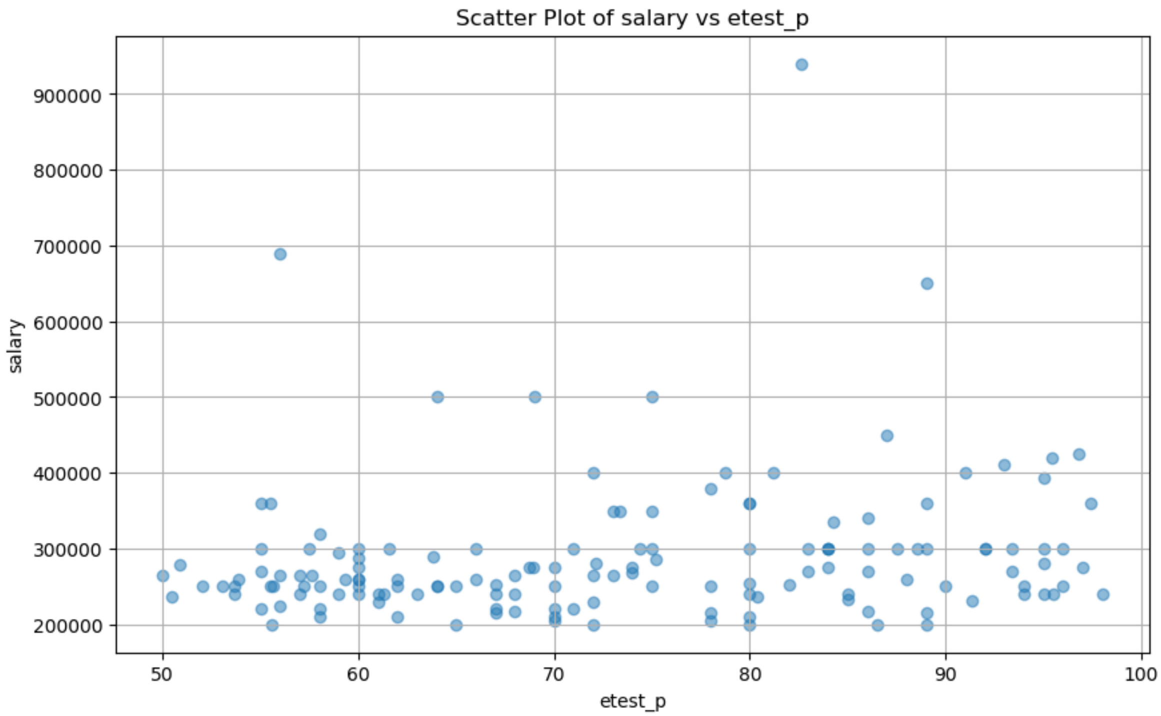
Out[52]:

	sl_no	ssc_p	hsc_p	degree_p	etest_p	mba_p	salary
sl_no	3870.000000	-52.641355	-58.106028	-40.413645	52.556168	8.102336	3.616177e+05
ssc_p	-52.641355	117.228377	60.348373	42.897137	37.659225	24.535952	2.877739e+04
hsc_p	-58.106028	60.348373	118.755706	34.819820	35.461678	22.555846	6.697772e+04
degree_p	-40.413645	42.897137	34.819820	54.151103	21.929469	17.272020	-1.173995e+04
etest_p	52.556168	37.659225	35.461678	21.929469	176.251018	16.886973	2.287876e+05
mba_p	8.102336	24.535952	22.555846	17.272020	16.886973	34.028376	9.624979e+04
salary	361617.668689	28777.386468	66977.716032	-11739.948520	228787.619507	96249.789024	8.734295e+09

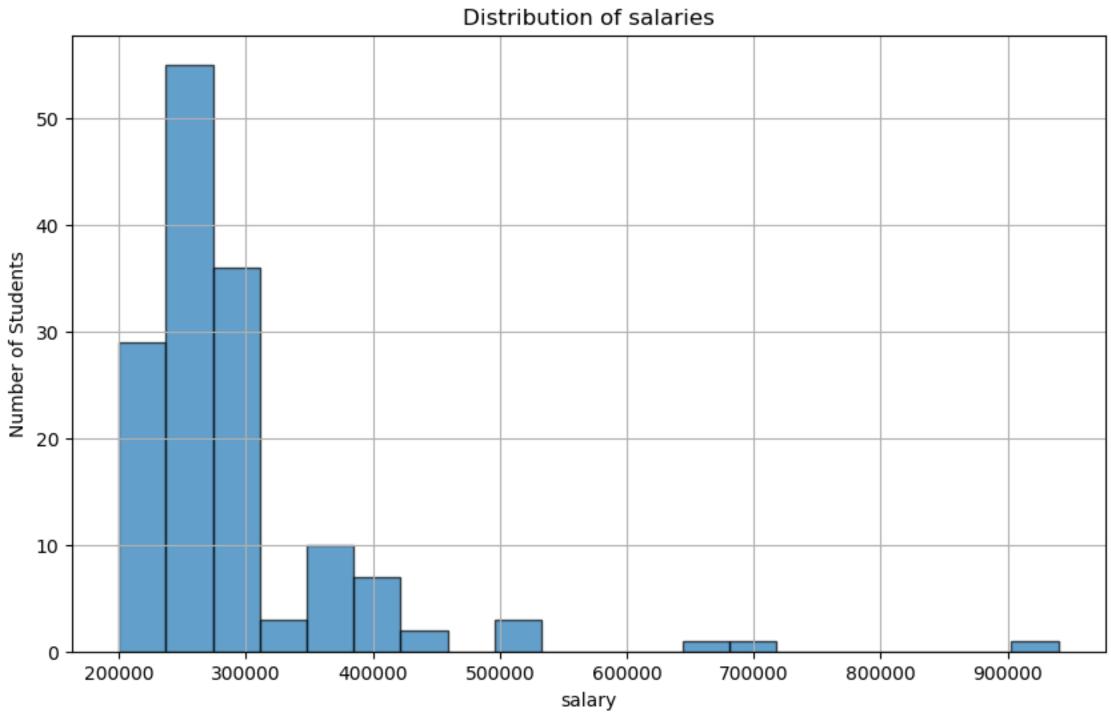
13) Plot any useful graph and explain it.

In [54]:
```
import pandas as pd
import matplotlib.pyplot as plt

# Load the dataset
df = pd.read_csv('Placement_Data_Full_Class.csv')

# Calculate the correlation matrix
correlation_matrix = df.corr()

# Extract the correlation of Salary with other columns
salary_correlation = correlation_matrix['salary']

# Find the parameter with the highest correlation with Salary (excluding Salary itself)
highest_correlation = salary_correlation.drop('salary').idxmax()
highest_value = salary_correlation[highest_correlation]

print(f"The parameter most highly correlated with salary is {highest_correlation} with a correlation of {highest_value:.2f}")

# Create a scatter plot
plt.figure(figsize=(10, 6))
plt.scatter(df[highest_correlation], df['salary'], alpha=0.5)
plt.title(f'Scatter Plot of salary vs {highest_correlation}')
plt.xlabel(highest_correlation)
plt.ylabel('salary')
plt.grid(True)
plt.show()
```

```
The parameter most highly correlated with salary is etest_p with a correlation of 0.18
```



In [2]:
```
import pandas as pd
import matplotlib.pyplot as plt

# Load the dataset
df = pd.read_csv('Placement_Data_Full_Class.csv')

# Plot a histogram of the Salary column
plt.figure(figsize=(10, 6))
plt.hist(df['salary'].dropna(), bins=20, edgecolor='black', alpha=0.7)
plt.title('Distribution of salaries')
plt.xlabel('salary')
plt.ylabel('Number of Students')
plt.grid(True)
plt.show()
```



Explanation Load the Dataset: The dataset is loaded into a DataFrame. Plot the Histogram: The histogram shows the distribution of the Salary column. plt.hist() is used to create the histogram. bins=20 specifies the number of bins (ranges) for the histogram. edgecolor='black' adds black borders to the bars for better visibility. alpha=0.7 makes the bars slightly transparent. The title, x-axis label, and y-axis label are set accordingly. plt.grid(True) adds a grid to the plot for better readability. Interpretation Histogram: The x-axis represents the salary ranges. The y-axis represents the number of students in each salary range. The height of each bar shows how many students earn within that salary range. This visualization helps us see the overall distribution of salaries, such as whether most students are earning lower, middle, or higher salaries. This histogram provides a clear view of how salaries are distributed among the students, making it easier to identify patterns or outliers in the salary data.

In [ ]: