1. Importar librerias y cargar datos

In [5]:
```
from sklearn import linear_model
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
experiencia = pd.Series([1.1,1.3,1.5,2.0,2.2,2.9,3.0,3.2,3.2,3.7,3.9,4.0,4.0,4.1,4.5,4.9,5.1,5.3,5.9,6.0,6.8,7.1,7.9,8.2,8.7,9.0,9.5,9.6,10.3,10.5])
experiencia
salario = pd.Series([39343.00, 46205.00, 37731.00, 43525.00, 39891.00, 56642.00, 60150.00, 54445.00, 64445.00, 57189.00, 63218.00, 55794.00, 56957.00, 57081.00, 61111.00, 67938.00, 66029.00, 83088.00, 81363.00, 93940.00, 91738.00
salario
df = pd.DataFrame({'Experiencia': experiencia, 'Salario': salario})
df
```

Out[5]:

	Experiencia	Salario
0	1.1	39343.0
1	1.3	46205.0
2	1.5	37731.0
3	2.0	43525.0
4	2.2	39891.0
5	2.9	56642.0
6	3.0	60150.0
7	3.2	54445.0
8	3.2	64445.0
9	3.7	57189.0
10	3.9	63218.0
11	4.0	55794.0
12	4.0	56957.0
13	4.1	57081.0
14	4.5	61111.0
15	4.9	67938.0
16	5.1	66029.0
17	5.3	83088.0
18	5.9	81363.0
19	6.0	93940.0
20	6.8	91738.0
21	7.1	98273.0
22	7.9	101302.0
23	8.2	113812.0
24	8.7	109431.0
25	9.0	105582.0
26	9.5	116969.0
27	9.6	112635.0
28	10.3	122391.0
29	10.5	121872.0

1. Análisis de datos

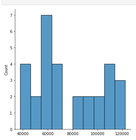
In [6]:
```
salario.isnull().sum()
```

Out[6]:
```
0
```

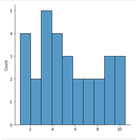
In [7]:
```
experiencia.isnull().sum()
```

Out[7]:
```
0
```

In [10]:
```
sns.displot(salario, bins=10)
plt.show()
```

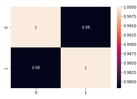


In [11]:
```
sns.displot(experiencia, bins=10)
plt.show()
```

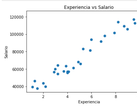


In [12]:
```
correlacion_matrix_salary = np.corrcoef(salario, experiencia)
sns.heatmap(data=correlacion_matrix_salary, annot = True)
```

Out[12]:
```
<AxesSubplot:>
```



In [13]:
```
plt.scatter(experiencia, salario)
plt.title('Experiencia vs Salario')
plt.xlabel('Experiencia')
plt.ylabel('Salario')
plt.show()
```



In [14]:
```
print('Maximo valor {}'.format(salario.max()))
print('Minimo valor {}'.format(salario.min()))
print('Mean valor {}'.format(salario.mean()))
print('Median valor {}'.format(salario.median()))
print('STD valor {}'.format(salario.std()))
```

```
Maximo valor 122391.0
Minimo valor 37731.0
Mean valor 76003.0
Median valor 65237.0
STD valor 27414.42978545823
```

1. Dividir los datos entre entrenamiento y test

20% -> Test

80% -> Train

In [19]:
```
x_experiencia = experiencia
y_salario = salario
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x_experiencia, y_salario, test_size = 0.2, random_state=42)
print(x_train.shape)
print(x_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(24,)
(6,)
(24,)
(6,)
```

1. Creando nuestro modelo

In [23]:
```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
#x_train = x_train.values.reshape(-1,1)
lin_model = LinearRegression()
lin_model.fit(x_train, y_train)
```

Out[23]:
```
LinearRegression()
```

1. Evaluando nuestro modelo

In [30]:
```
from sklearn.metrics import r2_score
#x_test = x_test.values.reshape(-1,1)
y_test_predict = lin_model.predict(x_test)
rmse = (np.sqrt(mean_squared_error(y_test, y_test_predict)))
print("The model performance for testing set")
print("--------------------------------------")
print('R2 score is {}'.format(r2))

print("--------------------------------------")
```

```
The model performance for testing set
--------------------------------------
R2 score is 0.9024461774180497
--------------------------------------
```

1. Pendiente de la recta y constante cuando x es igual a cero

y = mX + b

In [32]:
```
print('Coefficients: \n', lin_model.coef_)
print('Independent term: \n', lin_model.intercept_)
```

```
Coefficients:
 [9423.81532303]
Independent term:
 25321.583011776813
```

1. Recta de Regresion Lineal Simple

In [34]:
```
#experiencia = experiencia.values.reshape(-1,1)
predicted_data_salario = lin_model.predict(experiencia)
predicted_data_salario[0:5]
```

Out[34]:
```
array([35687.77986711, 37572.54293172, 39457.30599632, 44169.21365784,
       46053.97672244])
```

In [39]:
```
plt.scatter(experiencia, salario)
plt.plot(experiencia, predicted_data_salario, color ='red')
plt.title('E vs S')
plt.xlabel('E')
plt.ylabel('S')
plt.show()
```

1. Mostrar datos reales vs predicción

In [41]:
```
df1 = pd.DataFrame({'Real': salario, 'Predicción': predicted_data_salario})
df1
```

Out[41]:

	Real	Predicción
0	39343.0	35687.779867
1	46205.0	37572.542932
2	37731.0	39457.305996
3	43525.0	44169.213658
4	39891.0	46053.976722
5	56642.0	52850.647449
6	60150.0	53593.028981
7	54445.0	55477.792045
8	64445.0	55477.792045
9	57189.0	60188.699707
10	63218.0	62074.462772
11	55794.0	63016.844304
12	56957.0	63016.844304
13	57081.0	63958.225836
14	61111.0	67726.751965
15	67938.0	71496.278095
16	66029.0	73381.041159
17	83088.0	75267.804224
18	81363.0	80922.093418
19	93940.0	81864.474950
20	91738.0	89403.527208
21	98273.0	92230.671805
22	101302.0	99769.724064
23	113812.0	102596.868661
24	109431.0	107306.778322
25	105582.0	110135.920919
26	116969.0	114847.829581
27	112635.0	115790.210113
28	122391.0	122366.865039
29	121872.0	124271.643904

In [47]:
```
df1.head(15).plot(kind='bar')
plt.show()
```

In [49]:
```
df1.tail(15).plot(kind='bar')
plt.show()
```


In [ ]:
```
```