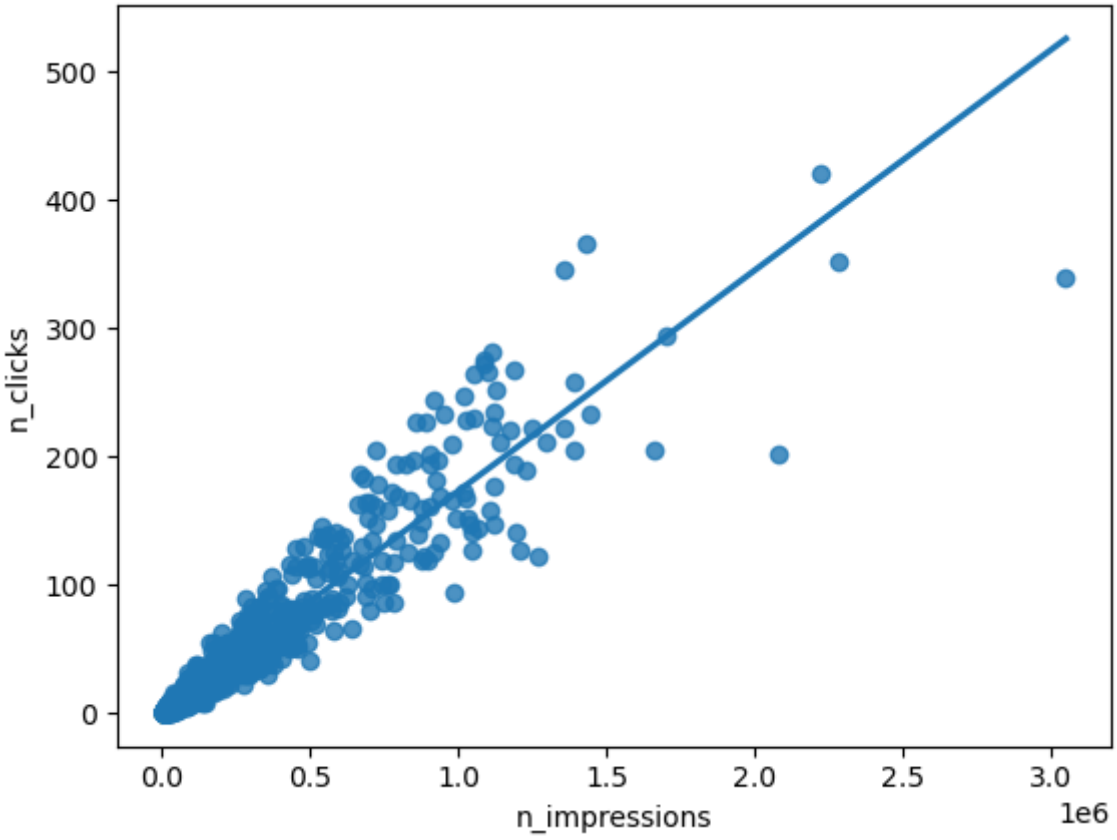
In [20]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from statsmodels.formula.api import ols
import numpy as np
```

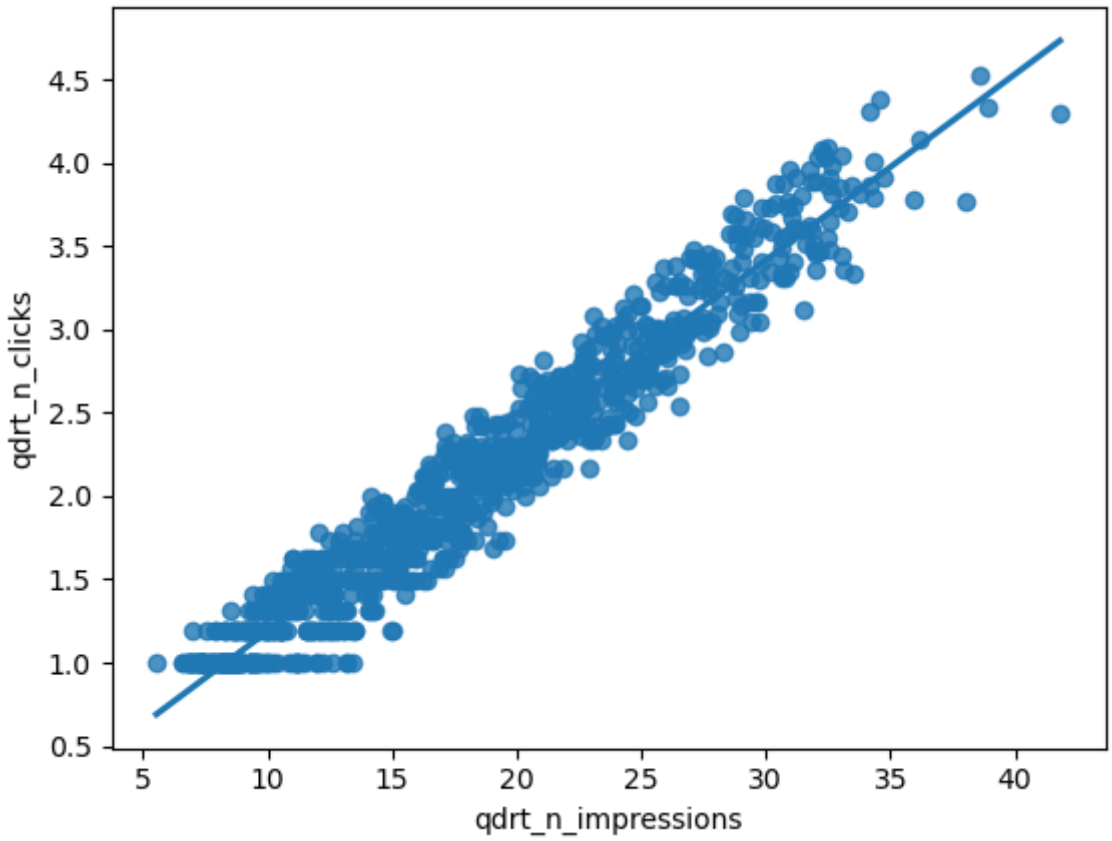
In [10]:
```
ad_conversion = pd.read_csv('ad_conversion.csv')
```

In [12]:
```
sns.regplot(x = "n_impressions" , y = "n_clicks" , data = ad_conversion, ci
plt.show()
```



In [19]:
```
ad_conversion["qdrt_n_impressions"] = ad_conversion["n_impressions"] ** 0.25
ad_conversion["qdrt_n_clicks"] = ad_conversion["n_clicks"] ** 0.25
sns.regplot(x = "qdrt_n_impressions" , y = "qdrt_n_clicks" , data=ad_convers
plt.show()
```

```
/var/folders/4j/bnvctt7152z6l5l6szd4m7wh0000gn/T/ipykernel_47058/3607702766.
py:1: PerformanceWarning: DataFrame is highly fragmented.  This is usually t
he result of calling `frame.insert` many times, which has poor performance.
Consider joining all columns at once using pd.concat(axis=1) instead. To get
a de-fragmented frame, use `newframe = frame.copy()`
  ad_conversion["qdrt_n_impressions"] = ad_conversion["n_impressions"] ** 0.
25
/var/folders/4j/bnvctt7152z6l5l6szd4m7wh0000gn/T/ipykernel_47058/3607702766.
py:2: PerformanceWarning: DataFrame is highly fragmented.  This is usually t
he result of calling `frame.insert` many times, which has poor performance.
Consider joining all columns at once using pd.concat(axis=1) instead. To get
a de-fragmented frame, use `newframe = frame.copy()`
  ad_conversion["qdrt_n_clicks"] = ad_conversion["n_clicks"] ** 0.25
```



In [21]:
```
mdl_click_vs_impression = ols("qdrt_n_clicks ~ qdrt_n_impressions", data = a
print(mdl_click_vs_impression.params)
```

```
Intercept              0.071748
qdrt_n_impressions     0.111533
dtype: float64
```

In [25]:
```
explanatory_data = pd.DataFrame({"qdrt_n_impressions": np.arange(0,3000000 ,
explanatory_data
```

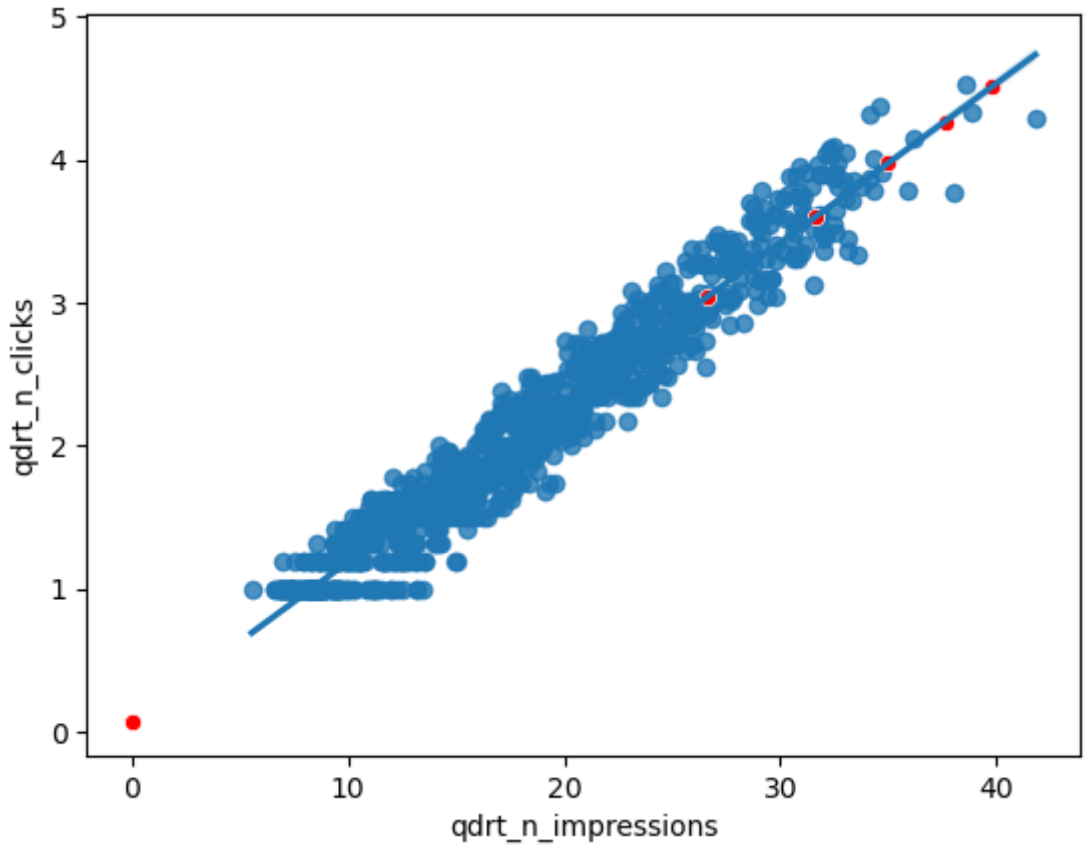
Out[25]:

	qdrt_n_impressions	n_impressions
0	0.000000	0
1	26.591479	500000
2	31.622777	1000000
3	34.996355	1500000
4	37.606031	2000000
5	39.763536	2500000

In [29]:
```
prediction_data = explanatory_data.assign(qdrt_n_clicks = mdl_click_vs_impre
print(prediction_data)
```

```
   qdrt_n_impressions  n_impressions  qdrt_n_clicks
0            0.000000              0       0.071748
1           26.591479         500000       3.037576
2           31.622777        1000000       3.598732
3           34.996355        1500000       3.974998
4           37.606031        2000000       4.266063
5           39.763536        2500000       4.506696
```

In [34]:
```
sns.regplot(x = "qdrt_n_impressions" , y = "qdrt_n_clicks" , data = ad_conve
sns.scatterplot(x = "qdrt_n_impressions" , y = "qdrt_n_clicks" , data = pred
plt.show()
```

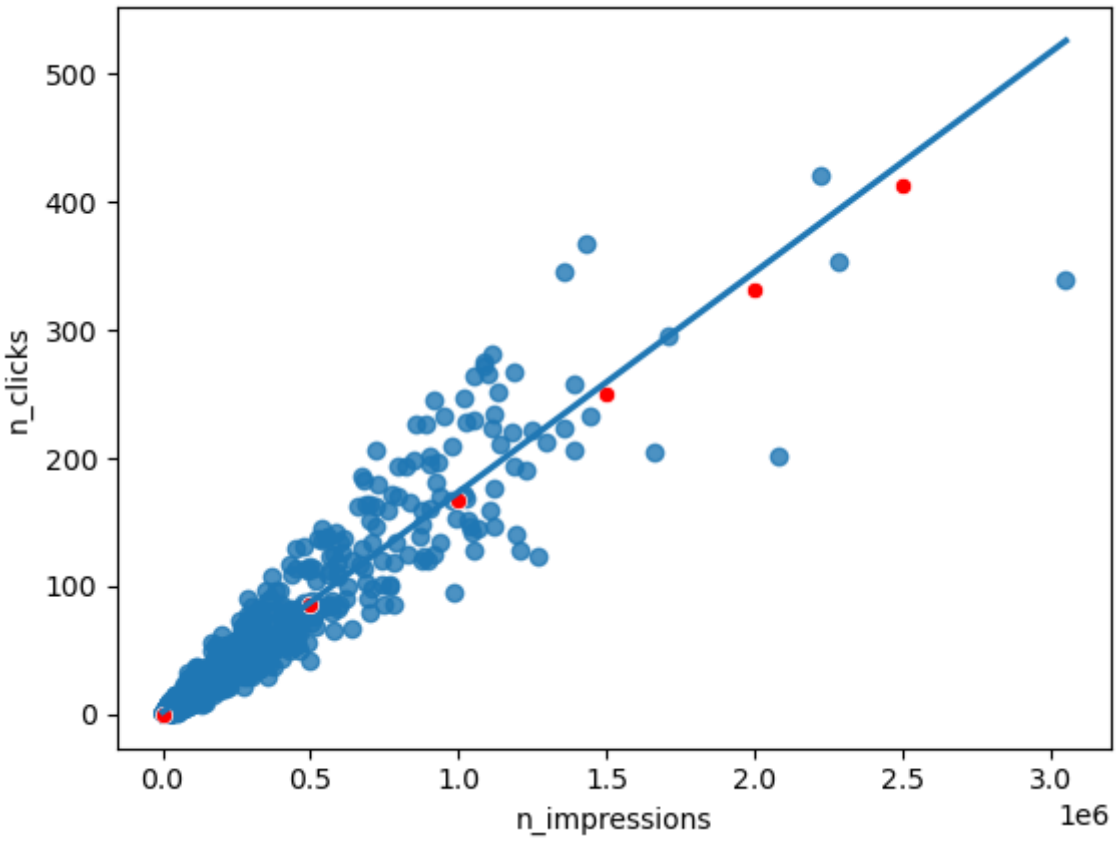


In [39]:
```
prediction_data['n_clicks'] = prediction_data["qdrt_n_clicks"] ** 4
prediction_data
```

Out[39]:

	qdrt_n_impressions	n_impressions	qdrt_n_clicks	n_clicks
0	0.000000	0	0.071748	0.000026
1	26.591479	500000	3.037576	85.135121
2	31.622777	1000000	3.598732	167.725102
3	34.996355	1500000	3.974998	249.659131
4	37.606031	2000000	4.266063	331.214159
5	39.763536	2500000	4.506696	412.508546

In [41]:
```
sns.regplot(x = "n_impressions" , y = "n_clicks" , data = ad_conversion, ci
sns.scatterplot(x = "n_impressions" , y = "n_clicks" , data = prediction_dat
plt.show()
```



In [47]:
```
print(mdl_click_vs_impression.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:          qdrt_n_clicks   R-squared:                       0.945
Model:                            OLS   Adj. R-squared:                  0.944
Method:                 Least Squares   F-statistic:                 1.590e+04
Date:                Sun, 10 Sep 2023   Prob (F-statistic):               0.00
Time:                        20:08:27   Log-Likelihood:                 193.90
No. Observations:                 936   AIC:                            -383.8
Df Residuals:                     934   BIC:                            -374.1
Df Model:                           1
Covariance Type:            nonrobust
==========================================================================
                         coef    std err          t      P>|t|      [0.025
0.975]
--------------------------------------------------------------------------
Intercept              0.0717      0.017      4.171      0.000       0.038
0.106
qdrt_n_impressions     0.1115      0.001    126.108      0.000       0.110
0.113
==============================================================================
Omnibus:                       11.447   Durbin-Watson:                   0.568
Prob(Omnibus):                  0.003   Jarque-Bera (JB):               10.637
Skew:                          -0.216   Prob(JB):                      0.00490
Kurtosis:                       2.707   Cond. No.                         52.1
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

In [46]:
```
print("R_SQUARE: " ,mdl_click_vs_impression.rsquared)
```

```
R_SQUARE:  0.9445272817143905
```

In [42]:
```
mse = mdl_click_vs_impression.mse_resid
print('mse: ', mse)
rse = np.sqrt(mse)
print("rse: ", rse)
```

```
mse:  0.03877213389297149
rse:  0.19690640896875725
```

In [ ]: