Electronic Supplementary Material (ESI) for Green Chemistry.

Supporting Information

1. Assessment on Parent amines and Related Structures.

To do a quick analysis when developing a chemical route for API production, we think about implementing CPCA approach earlier in stage. Except for detecting *N*-nitrosamine. two other structures as given in Figure S1 could also be considered in our algorithm. Hypothetically, structure in Figure S1 a) can be transformed to *N*-nitrosamine shown Figure S1 c) and Figure S1 b) structure could be transformed to *N*-nitrosamine given in Figure S1 d). The carcinogenic potency categorization approach will be applied to calculate the potency category from which we could have a general understanding of the risk of chemical intermediate by taking their potential *N*-nitrosamine into account.

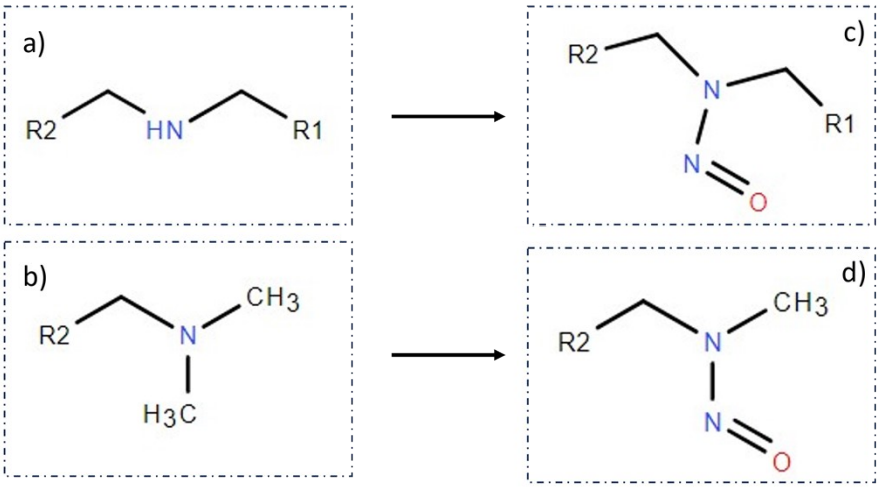
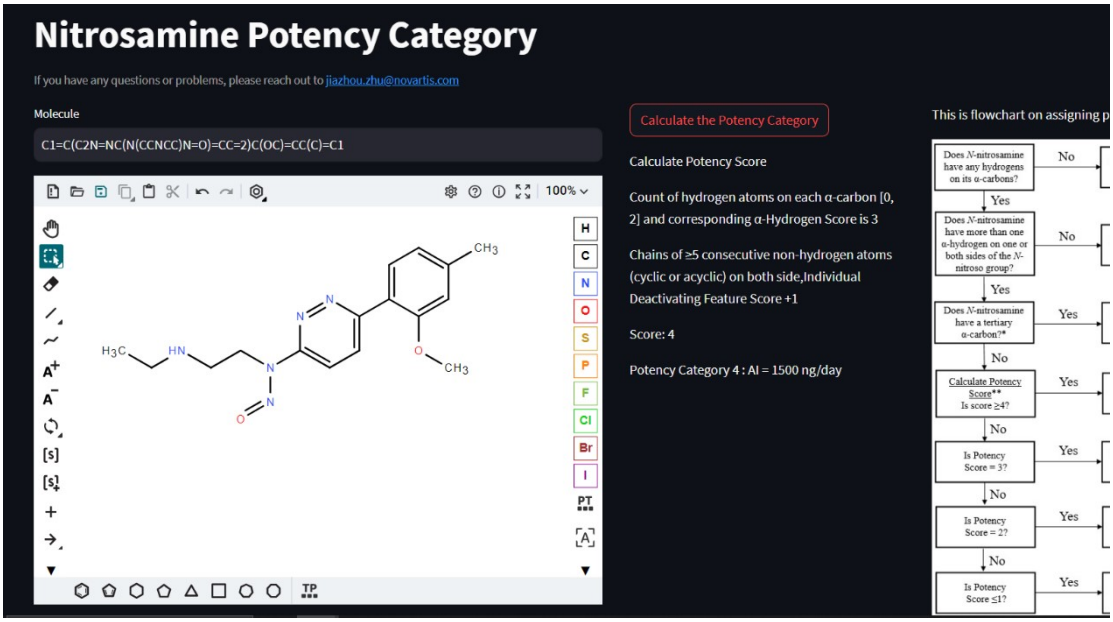


Figure S1. Transformation to *N*-nitrosamine

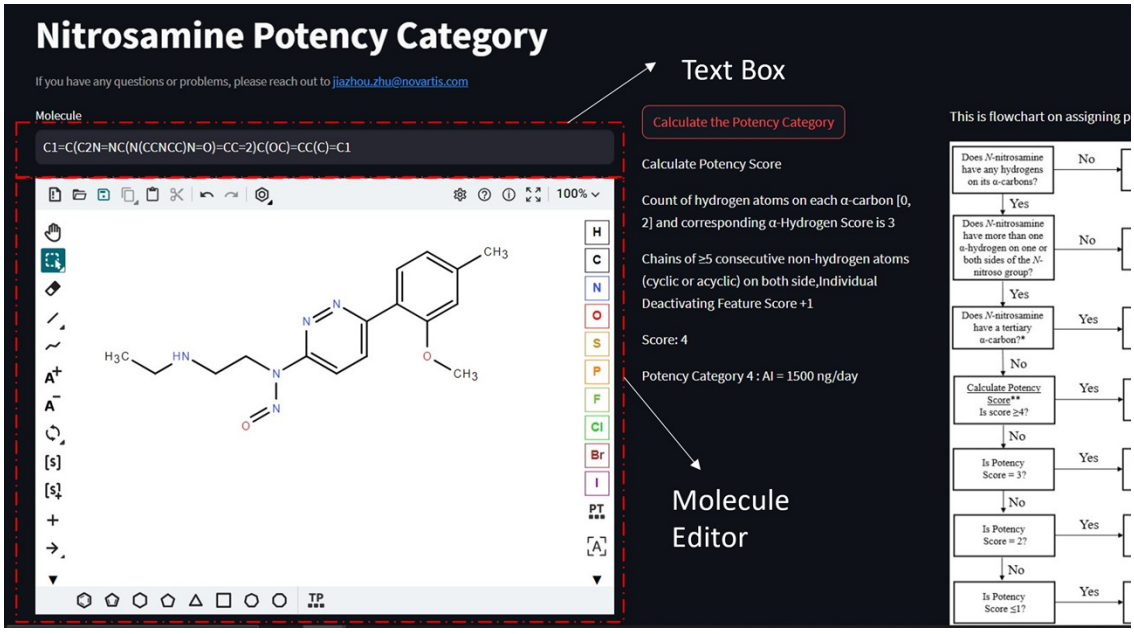
2. Brief Introduction on the Web Application

1. Once the web application is set up and ready, here gives the how in the interface

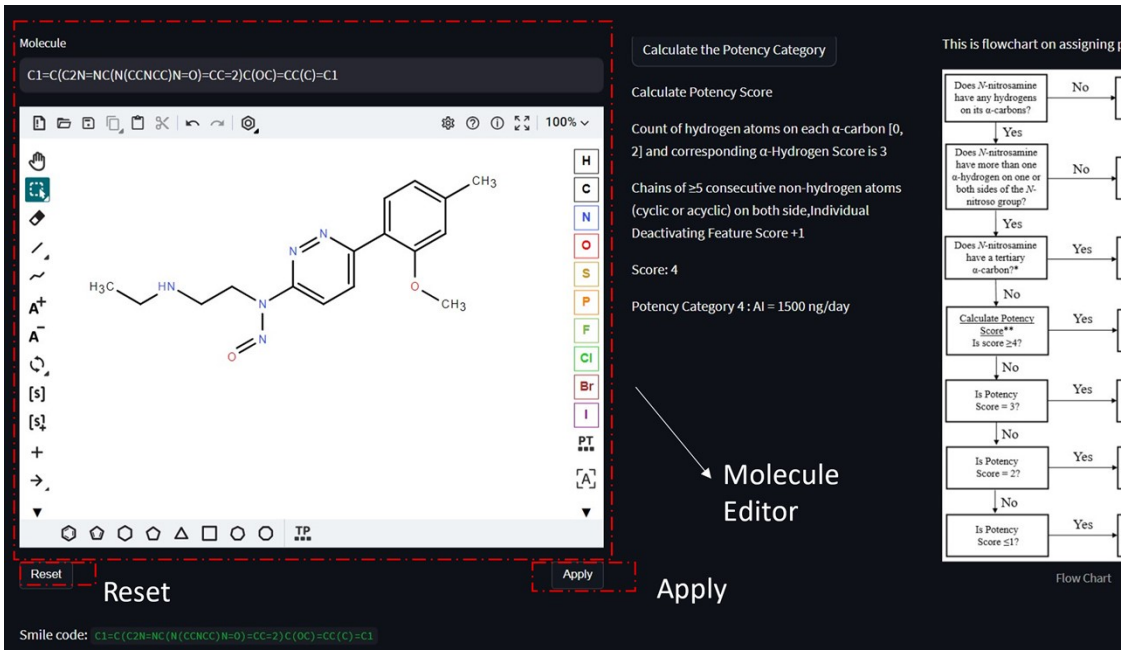


looks like

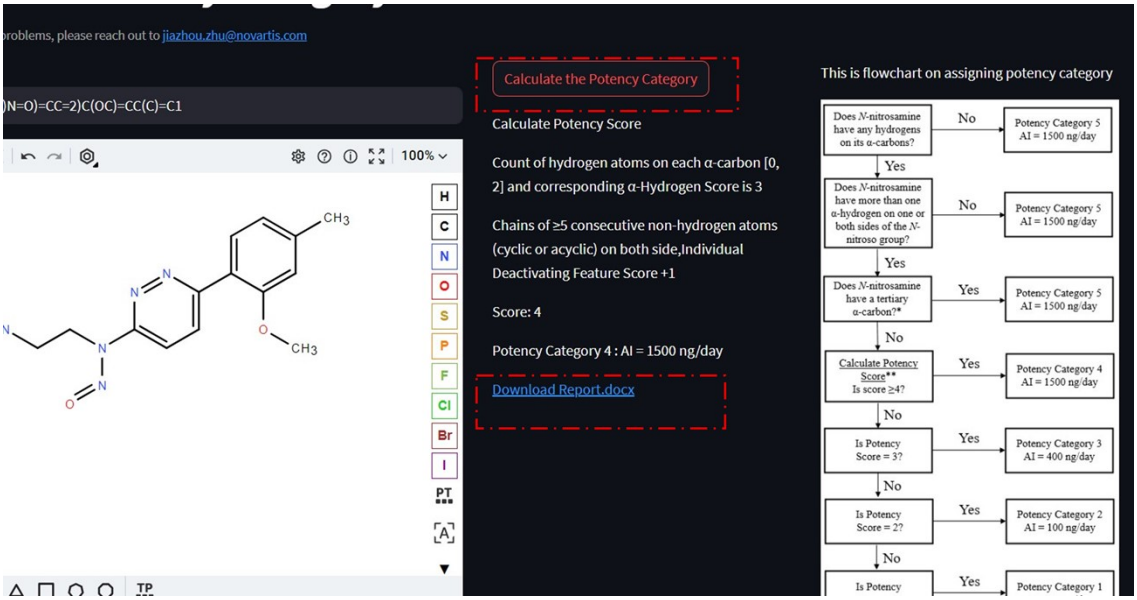
2. Smile string can be enter to either text box or the Molecule Editor



3. Once you have entered the molecule and you made further changes to the molecule, click apply button before you move on



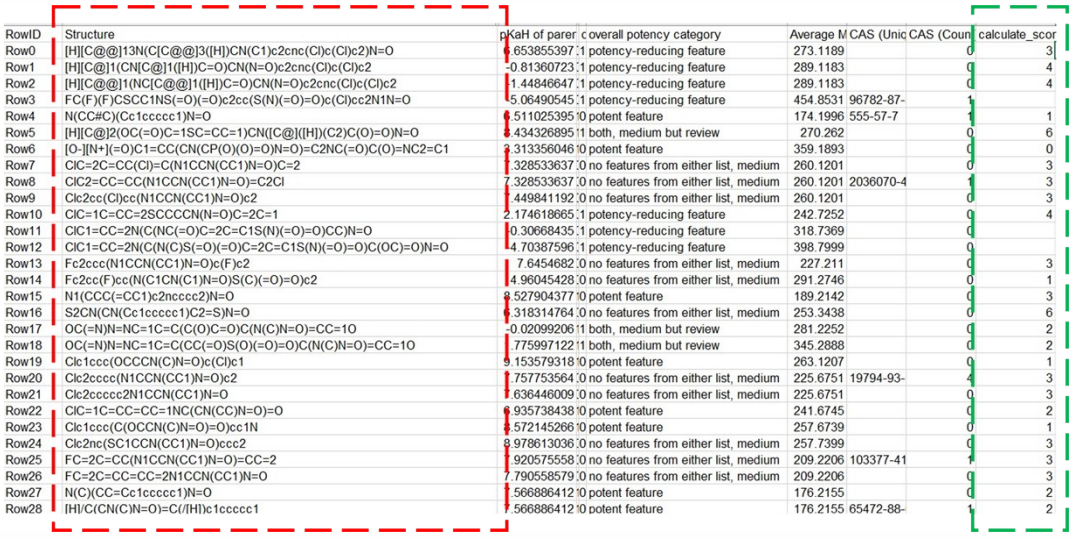
4. Once everything is ready, click on the 'calculate the potency category' button. The results will be given, and you can also download the report file. If you are not good with the template, you can edit the code to make your own template.



3. Data Preparation and Visualization

Raw code could be found through github link.

1. The smiles strings in red box will iteratively be calculated with our code, which is CPCA.py in GitHub link, calculated scores will be generated for each row and append to this excel sheet as given in the red box. A new csv file will be prepared used for further data analysis.



RowID	Structure	pKaH of parer	overall potency category	Average M	CAS (Unic	CAS (Coun	calculate_scor
Row0	[H][C@@]13N(C[C@@]3([H])CN(C1)c2cnc(Cl)c(Cl)c2)N=O	6.653855397	1 potency-reducing feature	273.1189		0	3
Row1	[H][C@]1(CN[C@]1([H])C=O)CN(N=O)c2cnc(Cl)c(Cl)c2	-0.81360723	1 potency-reducing feature	289.1183		0	4
Row2	[H][C@@]1(NC[C@@]1([H])C=O)CN(N=O)c2cnc(Cl)c(Cl)c2	-1.44846647	1 potency-reducing feature	289.1183		0	4
Row3	FC(F)(F)CSCC1NS(=O)(=O)c2cc(S(N)(=O)=O)c(Cl)cc2N1N=O	5.06490545	1 potency-reducing feature	454.8531	96782-87-	1	
Row4	N(CC#C)(Cc1ccccc1)N=O	6.511025395	0 potent feature	174.1996	555-57-7	1	1
Row5	[H][C@]2(OC(=O)C=1SC=CC=1)CN([C@]([H])(C2)C(O)=O)N=O	3.434326895	1 both, medium but review	270.262		0	6
Row6	[O-][N+](=O)C1=CC(CN(CP(O)(O)=O)N=O)=C2NC(=O)C(O)=NC2=C1	3.313356046	0 potent feature	359.1893		0	0
Row7	ClC=2C=CC(Cl)=C(N1CCN(CC1)N=O)C=2	7.328533637	0 no features from either list, medium	260.1201		0	3
Row8	ClC2=CC=CC(N1CCN(CC1)N=O)=C2Cl	7.328533637	0 no features from either list, medium	260.1201	2036070-4	1	3
Row9	Clc2cc(Cl)cc(N1CCN(CC1)N=O)c2	7.449841192	0 no features from either list, medium	260.1201		0	3
Row10	ClC=1C=CC=2SCCCCN(N=O)C=2C=1	2.174618665	1 potency-reducing feature	242.7252		0	4
Row11	ClC1=CC=2N(C(NC(=O)C=2C=C1S(N)(=O)=O)CC)N=O	0.30668435	1 potency-reducing feature	318.7369		0	
Row12	ClC1=CC=2N(C(N(C)S(=O)(=O)C=2C=C1S(N)(=O)=O)C(OC)=O)N=O	4.70387596	1 potency-reducing feature	398.7999		0	
Row13	Fc2ccc(N1CCN(CC1)N=O)c(F)c2	7.6454682	0 no features from either list, medium	227.211		0	3
Row14	Fc2cc(F)cc(N(C1CN(C1)N=O)S(C)(=O)=O)c2	4.96045428	0 no features from either list, medium	291.2746		0	1
Row15	N1(CCC(=CC1)c2ncccc2)N=O	8.527904377	0 potent feature	189.2142		0	3
Row16	S2CN(CN(Cc1ccccc1)C2=S)N=O	6.318314764	0 no features from either list, medium	253.3438		0	6
Row17	OC(=N)N=NC=1C=C(C(O)C=O)C(N(C)N=O)=CC=1O	-0.02099206	1 both, medium but review	281.2252		0	2
Row18	OC(=N)N=NC=1C=C(CC(=O)S(O)(=O)=O)C(N(C)N=O)=CC=1O	1.775997122	1 both, medium but review	345.2888		0	2
Row19	Clc1ccc(OCCCN(C)N=O)c(Cl)c1	9.153579318	0 potent feature	263.1207		0	1
Row20	Clc2cccc(N1CCN(CC1)N=O)c2	7.757753564	0 no features from either list, medium	225.6751	19794-93-	4	3
Row21	Clc2ccccc2N1CCN(CC1)N=O	7.636446009	0 no features from either list, medium	225.6751		0	3
Row22	ClC=1C=CC=CC=1NC(CN(CC)N=O)=O	6.935738438	0 potent feature	241.6745		0	2
Row23	Clc1ccc(C(OCCN(C)N=O)=O)cc1N	8.572145266	0 potent feature	257.6739		0	1
Row24	Clc2nc(SC1CCN(CC1)N=O)ccc2	8.978613036	0 no features from either list, medium	257.7399		0	3
Row25	FC=2C=CC(N1CCN(CC1)N=O)=CC=2	7.920575558	0 no features from either list, medium	209.2206	103377-41	1	3
Row26	FC=2C=CC=CC=2N1CCN(CC1)N=O	7.790558579	0 no features from either list, medium	209.2206		0	3
Row27	N(C)(CC=Cc1ccccc1)N=O	7.566886412	0 potent feature	176.2155		0	2
Row28	[H]/C(CN(C)N=O)=C(/[H])c1ccccc1	7.566886412	0 potent feature	176.2155	65472-88-	1	2

2. The calculated scores will be assigned with potency categories based on the CPCA rules.

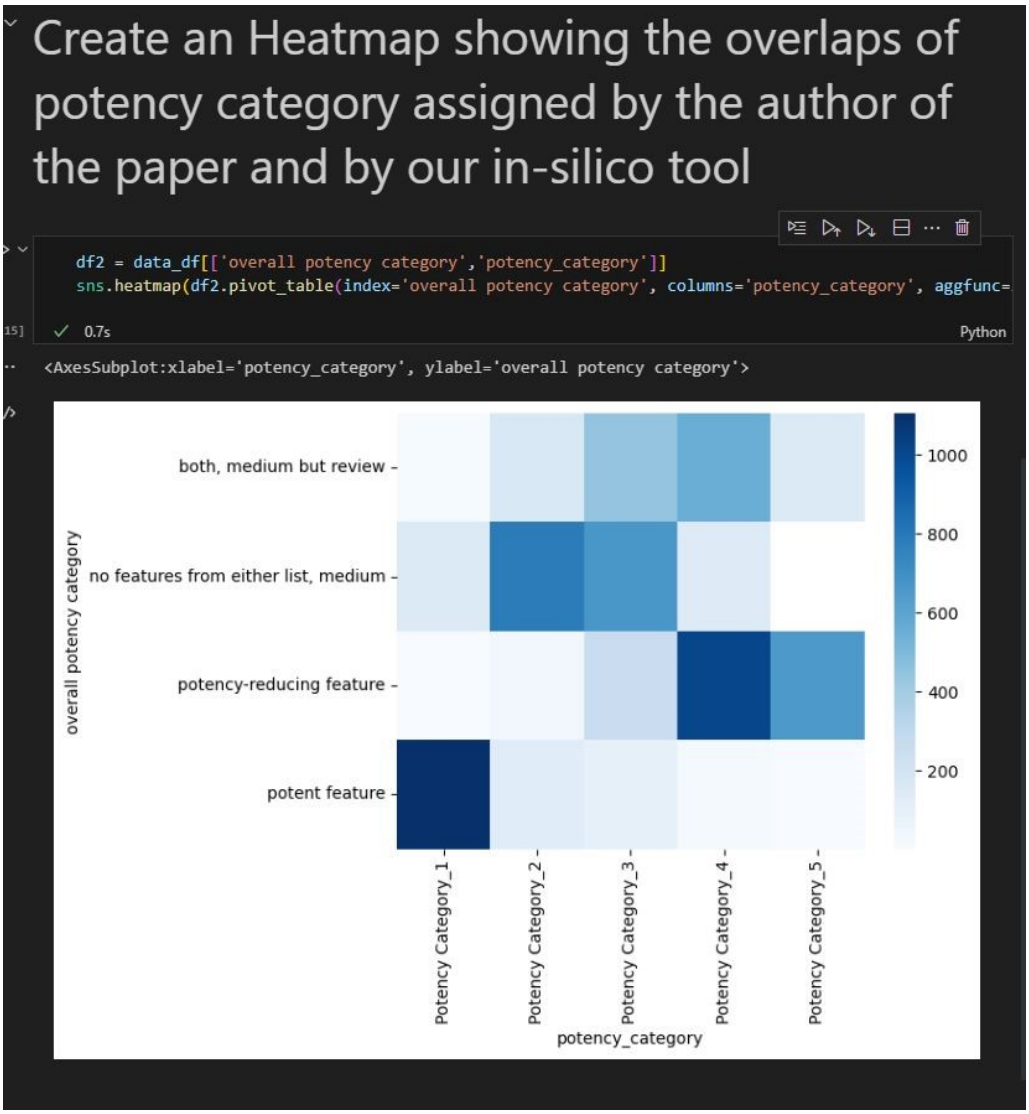
```
category_list = []
for i in score_list:
    if not i:
        category_list.append('Potency Category_5')
    elif np.isnan(float(i)):
        category_list.append('Potency Category_5')
    elif int(i) >= 4:
        category_list.append('Potency Category_4')
    elif int(i) == 3:
        category_list.append('Potency Category_3')
    elif int(i) == 2:
        category_list.append('Potency Category_2')
    elif int(i) <= 1:
        category_list.append('Potency Category_1')
```

✓ 0.0s

```
data_df['potency_category'] = category_list
```

✓ 0.0s

3. Once the categories are assigned to each molecule, an heat map will be created based on the two columns which are ‘overall potency category’ and ‘potency_category’. The previous one is the raw category assigned by Schlingemann,Joerg,et al. The latter one is based on the CPCA rules.

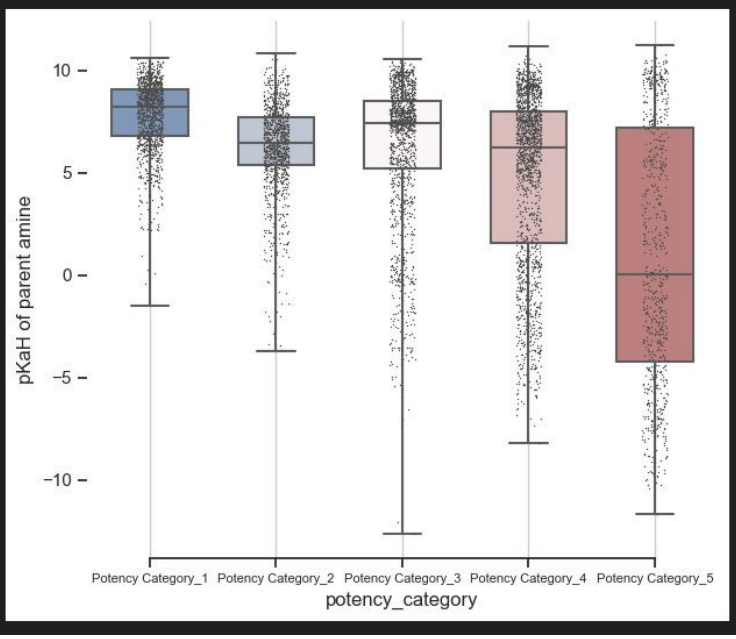


4. A boxplot along with the striplot are also provided using seaborn package in python to visualize the distribution of the pka values of their parent amines under different potency categories. 75%, 50% and 25% percentile of the pka values are given in plot. The detailed code is given as below.

```python
import seaborn as sns
import matplotlib.pyplot as plt

sns.set_theme(style="ticks")

# Initialize the figure with a logarithmic x axis
f, ax = plt.subplots(figsize=(7, 6))

# Plot the orbital period with horizontal boxes
sns.boxplot(x='potency_category', y='pKaH of parent amine', data=data_df,
            whis=[0, 100], width=.6, palette="vlag",order=['Potency Category_1', 'Potency Category
'Potency Category_4','Potency Category_5'])

# Add in points to show each observation
sns.stripplot(x='potency_category', y='pKaH of parent amine', data=data_df,
              size=1, color=".3", linewidth=0,order=['Potency Category_1', 'Potency Category_2', '
'Potency Category_4','Potency Category_5'])

# Tweak the visual presentation
ax.xaxis.grid(True)
ax.set(ylabel="pKaH of parent amine")
sns.despine(trim=True, left=True)

ax.set_xticklabels(ax.get_xticklabels(), fontsize=8)
```

Python

```
[Text(0, 0, 'Potency Category_1'),
 Text(1, 0, 'Potency Category_2'),
 Text(2, 0, 'Potency Category_3'),
 Text(3, 0, 'Potency Category_4'),
 Text(4, 0, 'Potency Category_5')]
```

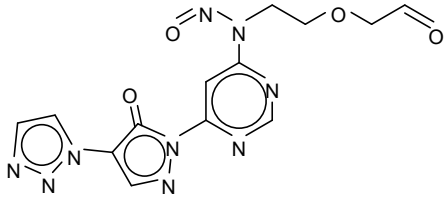


4. Unrecognized structures in the dataset

16 unrecognized structures are given as below, invalid pattern can be found in all of given structures (where the structures are generated in ChemDraw Software based on the SMILES string)

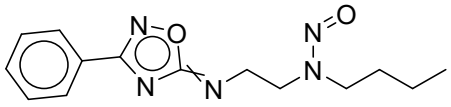
Row479

O(CCN(N=O)c1ncnc(c1)-n3ncc(-n2nncc2)c3=O)CC=O



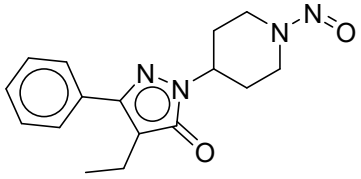
Row813

N(CCCC)(CCN=c1nc(no1)-c2ccccc2)N=O



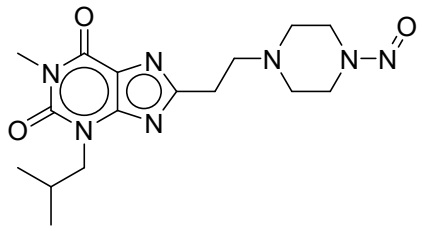
Row1354

N1(CCC(CC1)n2nc(c(CC)c2=O)-c3ccccc3)N=O



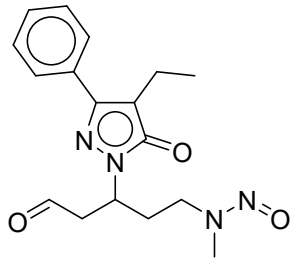
Row1489

N1(CCN(CC1)N=O)CCc2nc3n(CC(C)C)c(=O)n(C)c(=O)c3n2



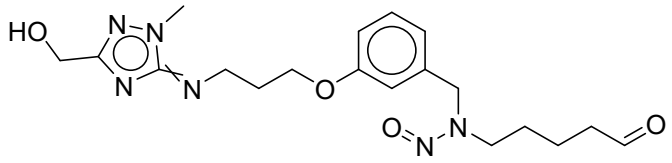
Row1759

N(C)(CCC(CC=O)n1nc(c(CC)c1=O)-c2ccccc2)N=O



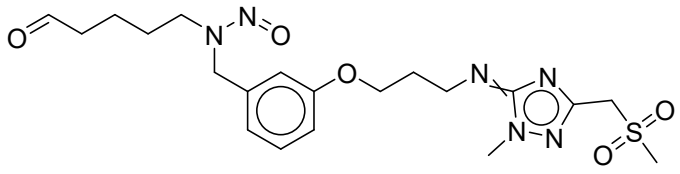
Row2721

OCc2nn(C)c(=NCCCOc1cccc(CN(CCCCC=O)N=O)c1)n2



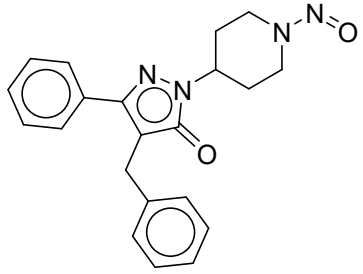
Row3161

O(CCCN=c1nc(CS(C)(=O)=O)nn1C)c2cccc(CN(CCCCC=O)N=O)c2



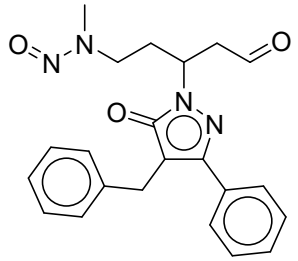
Row3299

N1(CCC(CC1)n4nc(-c2ccccc2)c(Cc3ccccc3)c4=O)N=O



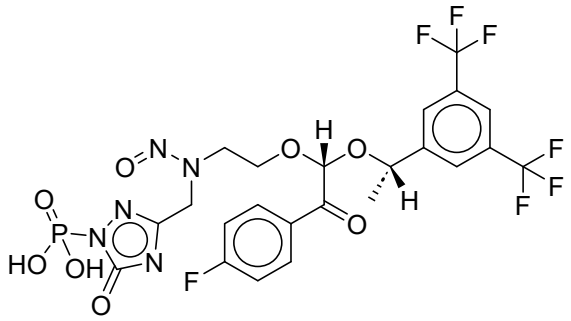
Row3736

N(C)(CCC(CC=O)n2nc(c(Cc1ccccc1)c2=O)-c3ccccc3)N=O



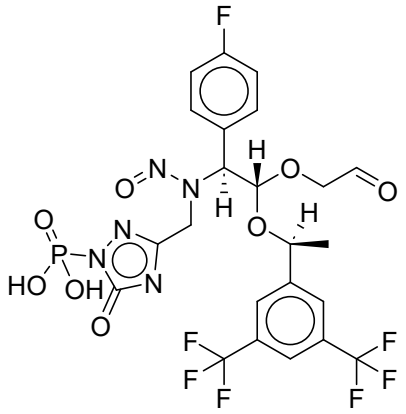
Row3996

[H][C@@](OCCN(Cc1nn(P(O)(O)=O)c(=O)n1)N=O)(O[C@@]([H])(C)c2cc(C(F)(F)F)cc(C(F)(F)F)c2)C(=O)c3ccc(F)cc3



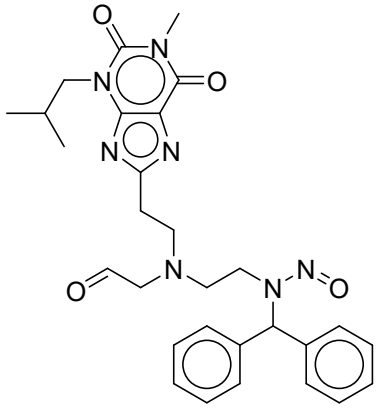
Row3997

[H][C@@](OCC=O)(O[C@@]([H])(C)c1cc(C(F)(F)F)cc(C(F)(F)F)c1)[C@@]([H])(N(Cc2nn(P(O)(O)=O)c(=O)n2)N=O)c3ccc(F)cc3



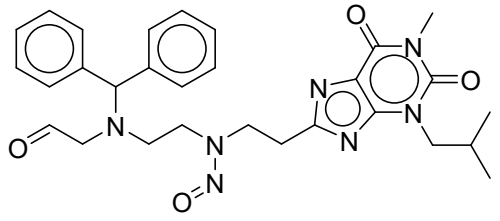
Row5285

N(CCN(C(c1ccccc1)c2ccccc2)N=O)(CCc3nc4n(CC(C)C)c(=O)n(C)c(=O)c4n3)CC=O



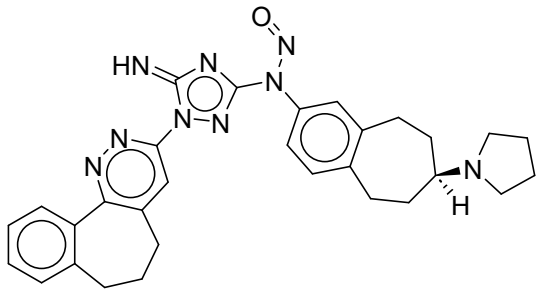
Row5286

N(CCN(CCc1nc2n(CC(C)C)c(=O)n(C)c(=O)c2n1)N=O)(CC=O)C(c3ccccc3)c4ccccc4



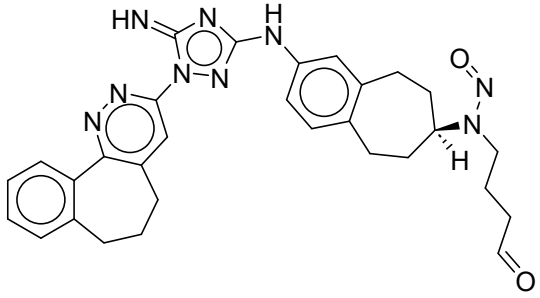
Row5381

[H][C@]6(N1CCCC1)CCc7ccc(N(N=O)c5nn(c4nnc3c(CCCc2ccccc23)c4)c(=N)n5)cc7C
C6



Row5382

[H][C@]5(N(CCCC=O)N=O)CCc6ccc(Nc4nn(c3nnc2c(CCCc1ccccc12)c3)c(=N)n4)cc6C
C5



Row6365

[O-][n+]1c(N)cc(N(CCCCC=O)N=O)nc1=N