```
def get_recent_cves(days=2):
```

pubEndDate=now,

```
now = datetime.now()
```

```
cvssV3Severity='CRITICAL'
```

pubStartDate=days_ago,

```
@lru_cache(maxsize=None)
```

```
cves = nvdlib.searchCVE(
```

```
days_ago = now - timedelta(days=days)
```


return

for cve in cves

```
(cve.id, _description(cve), _cvss_score(cve))
```

```
def __init__(self, options, columns):
```

```
def execute(self, quals, columns):
```

```
super().__init__(options, columns)
```

```
class CVEData(ForeignDataWrapper):
```

```
cves = get_recent_cves()
```

return cves

```
def __init__(self, options, columns):
```

```
class NpNormal(ForeignDataWrapper):
```

```
super().__init__(options, columns)
```

```
for qual in quals if qual.operator == '='
```

return {

```
def _quals_to_dict(self, quals):
```

```
qual.field_name: int(qual.value)
```

```
self.size = int(options.get('size', 100))
```

```
self.std = int(options.get('std', 1))
```

```
self.mean = int(options.get('mean', 0))
```

```
def execute(self, quals, columns):
```

```
std = _quals_dict.get('std', self.std)
```

```
yield idx, smpl, mean, std, size
```

```
_quals_dict = self._quals_to_dict(quals)
```

):

```
size = _quals_dict.get('size', self.size)
```

```
np.random.normal(loc=mean, scale=std, size=size)
```

```
for idx, smpl in enumerate(
```

```
mean = _quals_dict.get('mean', self.mean)
```

```
def _quals_to_dict(self, quals):
```