In [1]:
```
from sqlalchemy import create_engine
import pandas as pd
import matplotlib.pyplot as plt
import psycopg2
import seaborn as sns
import numpy as np
import os

from pathlib import Path
import nbconvert
import base64
import nbformat
```

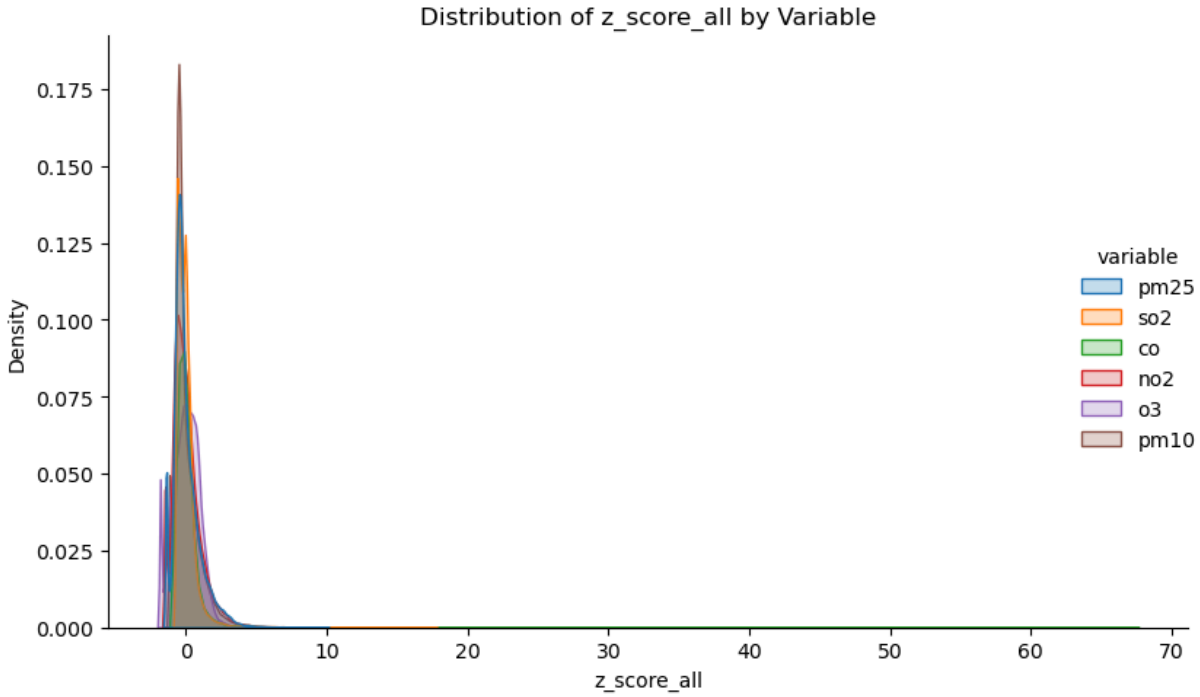
In [2]:
```
DATABASE_TYPE = 'postgresql'
DBAPI = 'psycopg2'
HOST = '127.0.0.1'
USER = 'postgres'
PASSWORD = 'password'
DATABASE = 'PM25'
PORT = 5432


engine = create_engine(f"{DATABASE_TYPE}+{DBAPI}://{USER}:{PASSWORD}@{HOST}:{PORT}/
```

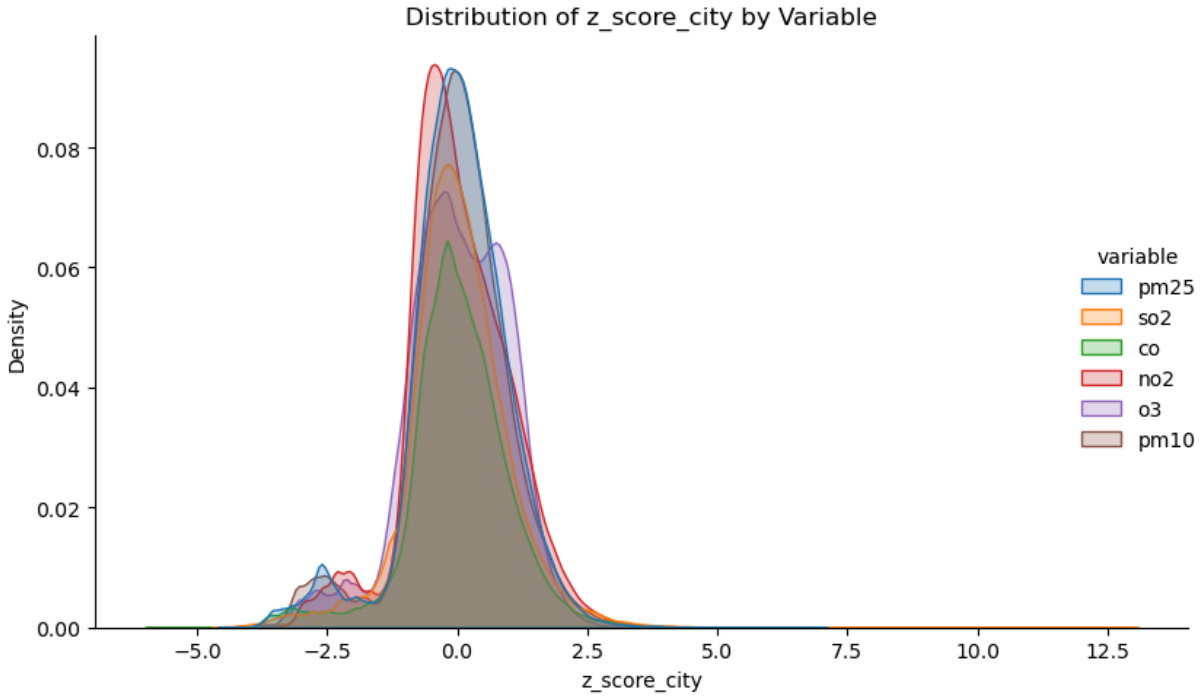
In [15]:
```
sql_file_path = os.path.join("..", "..", "03_SQL_queries", "01_data_exploration", "


with open(sql_file_path, 'r') as file:
    query = file.read()

df = pd.read_sql(query, engine)


print(df.head())
print(df.info())
```

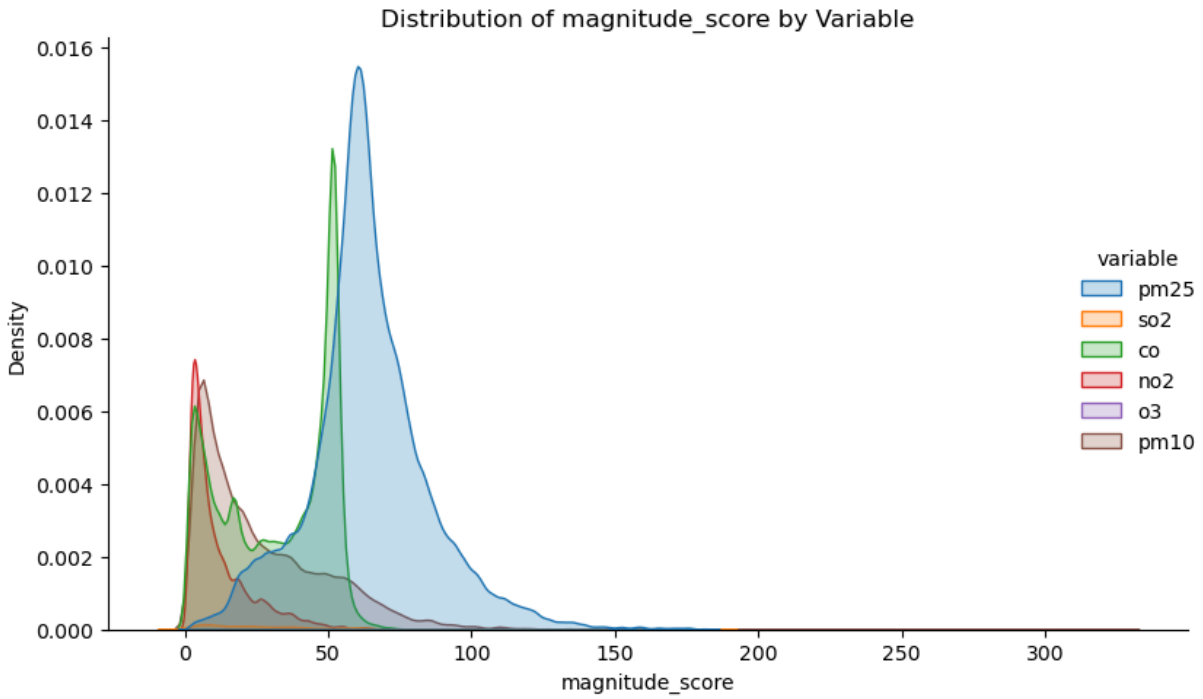
```
         date     city variable   min   max  median  30_day_rolling_avg  \
0  2019-01-06  Quilpué     pm25  46.0  63.0    61.0               64.33
1  2019-01-31  Quilpué     pm25  52.0  55.0    53.0               52.75
2  2019-01-30  Quilpué     pm25  45.0  60.0    52.0               53.42
3  2019-01-04  Quilpué     pm25  58.0  75.0    64.0               67.25
4  2019-01-05  Quilpué     pm25  45.0  59.0    56.0               65.00

   z_score_all  z_score_city  magnitude_score
0         0.45          0.81             69.0
1         0.11          0.00             69.0
2         0.13          0.05             69.0
3         0.53          1.02             69.0
4         0.47          0.86             69.0
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4127791 entries, 0 to 4127790
Data columns (total 10 columns):
 #   Column              Dtype
---  ------              -----
 0   date                object
 1   city                object
 2   variable            object
 3   min                 float64
 4   max                 float64
 5   median              float64
 6   30_day_rolling_avg  float64
 7   z_score_all         float64
 8   z_score_city        float64
 9   magnitude_score     float64
dtypes: float64(7), object(3)
memory usage: 314.9+ MB
None
```

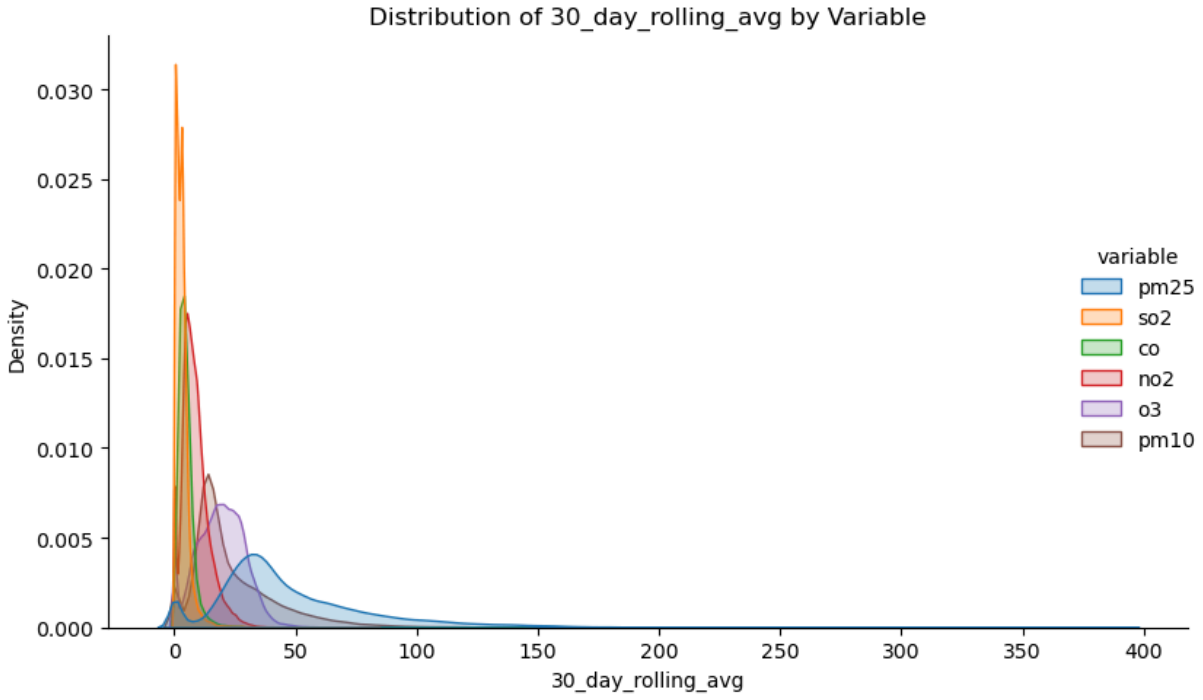
In [7]:
```
cal_cols = ["z_score_all", "z_score_city", "magnitude_score","30_day_rolling_avg"]
val_cols = ["min", "max", "median"]
```

In [10]:
```
for col in cal_cols:

    sns.displot(
        data=df,
        x=f"{col}",
        hue="variable",
        kind="kde", # use KDE to get a clearer view of the overall distribution
        fill=True,  # fills under the KDE curve for easier readability
        aspect=1.5
    )
    plt.title(f"Distribution of {col} by Variable")
    plt.tight_layout()
    plt.show()
```

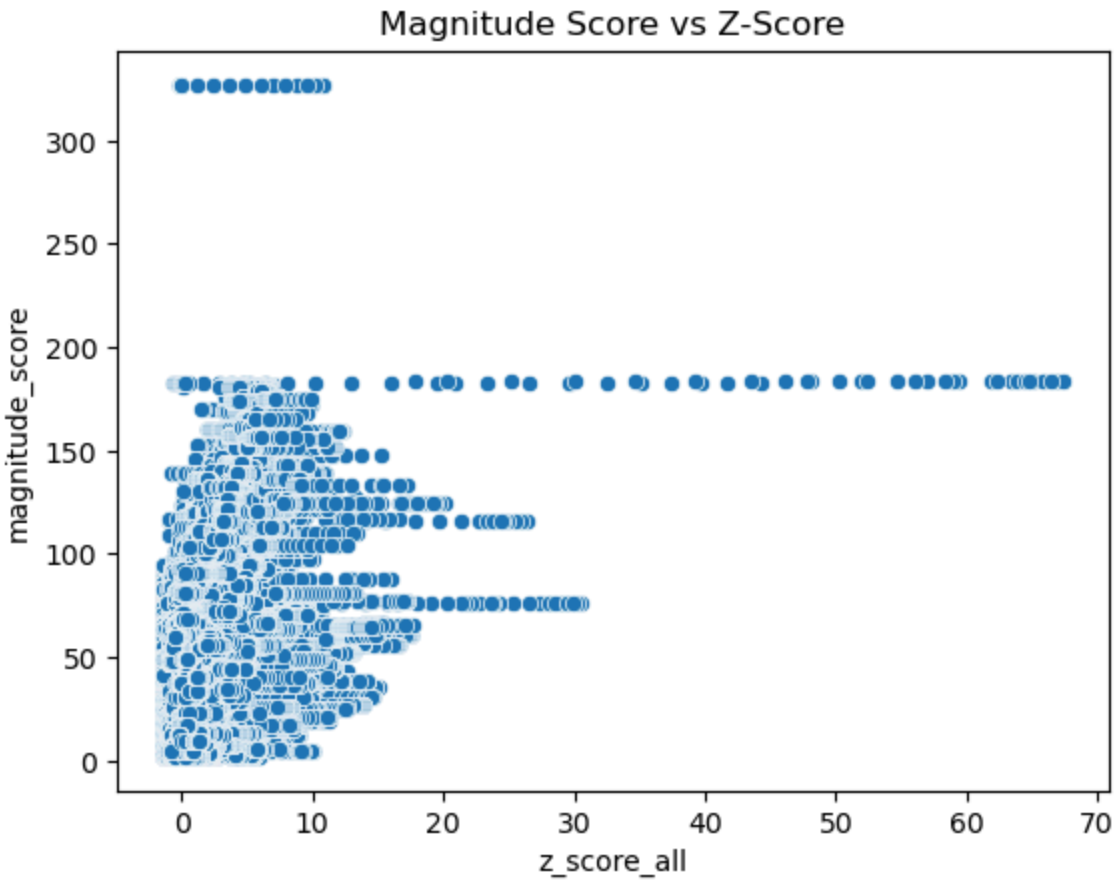

Distribution of z_score_all by Variable
Density
z_score_all
variable
pm25
so2
co
no2
o3
pm10
0.175
0.150
0.125
0.100
0.075
0.050
0.025
0.000
0
10
20
30
40
50
60
70


Distribution of z_score_city by Variable
Density
z_score_city
variable
pm25
so2
co
no2
o3
pm10
0.08
0.06
0.04
0.02
0.00
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0
12.5

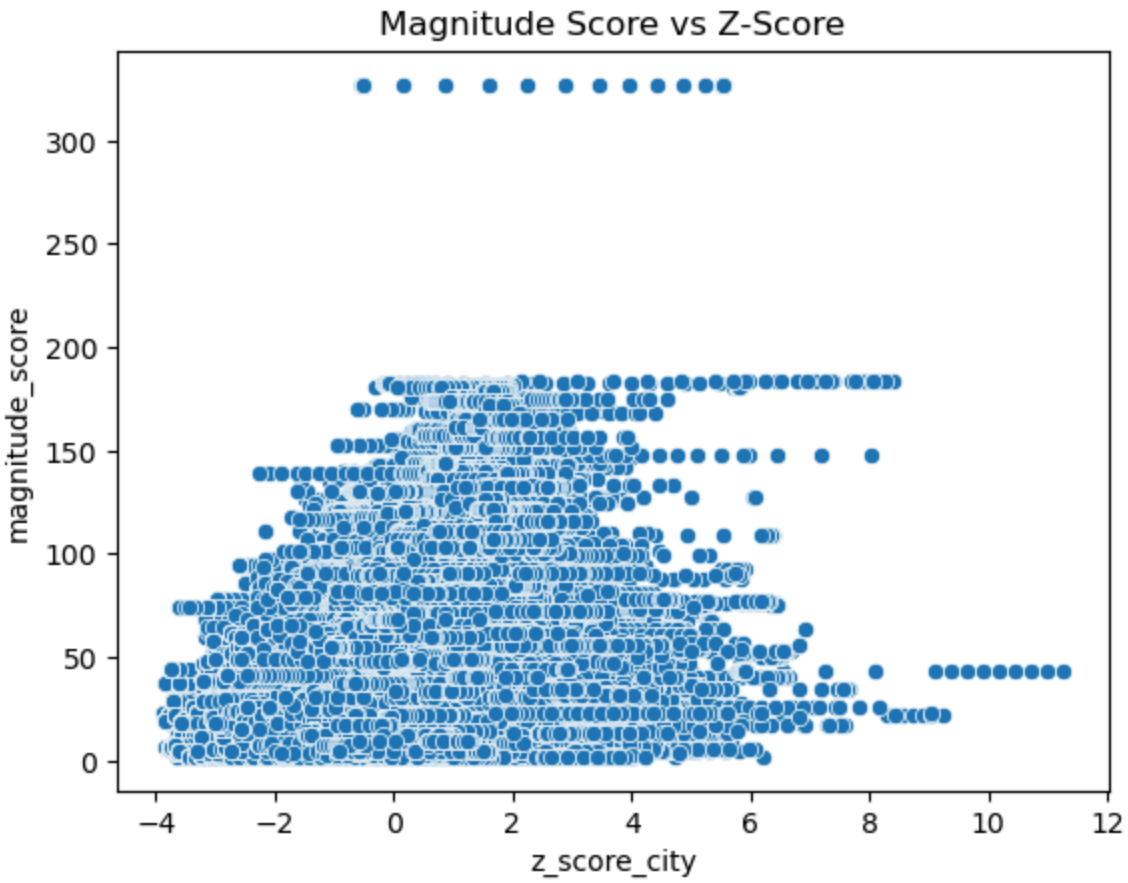




In [ ]:
```
sns.scatterplot(data=df, x="z_score_all", y="magnitude_score")
plt.title("Magnitude Score vs Z-Score")
plt.show()
```



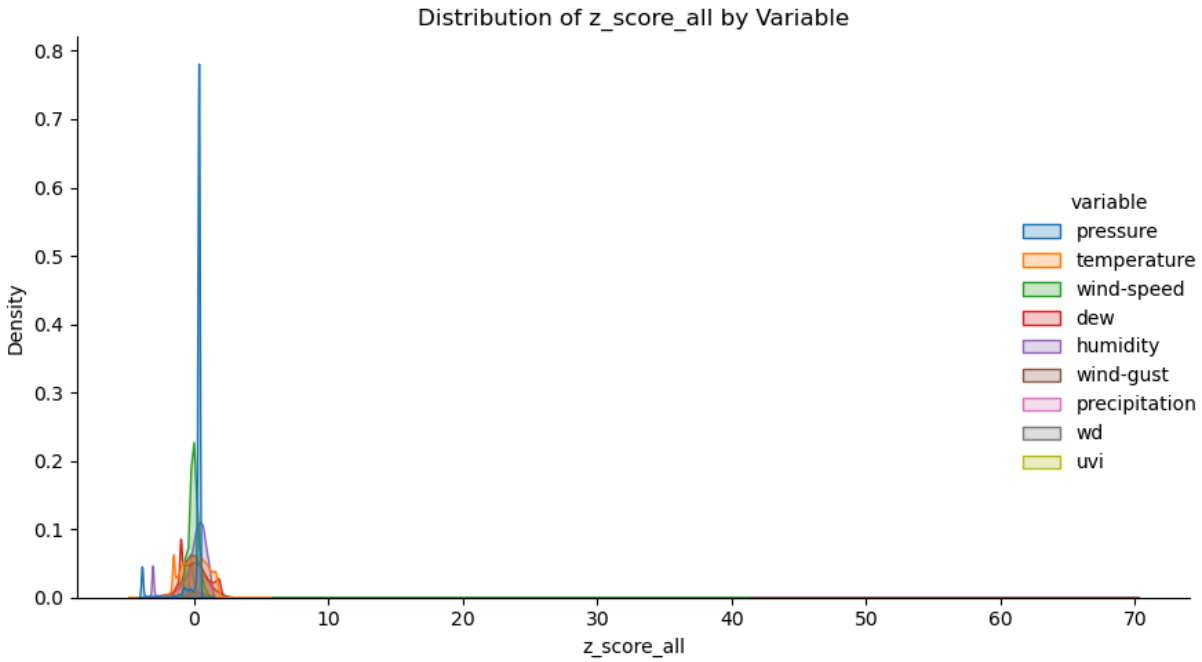
In [ ]:
```
sns.scatterplot(data=df, x="z_score_city", y="magnitude_score")
plt.title("Magnitude Score vs Z-Score")
plt.show()
```

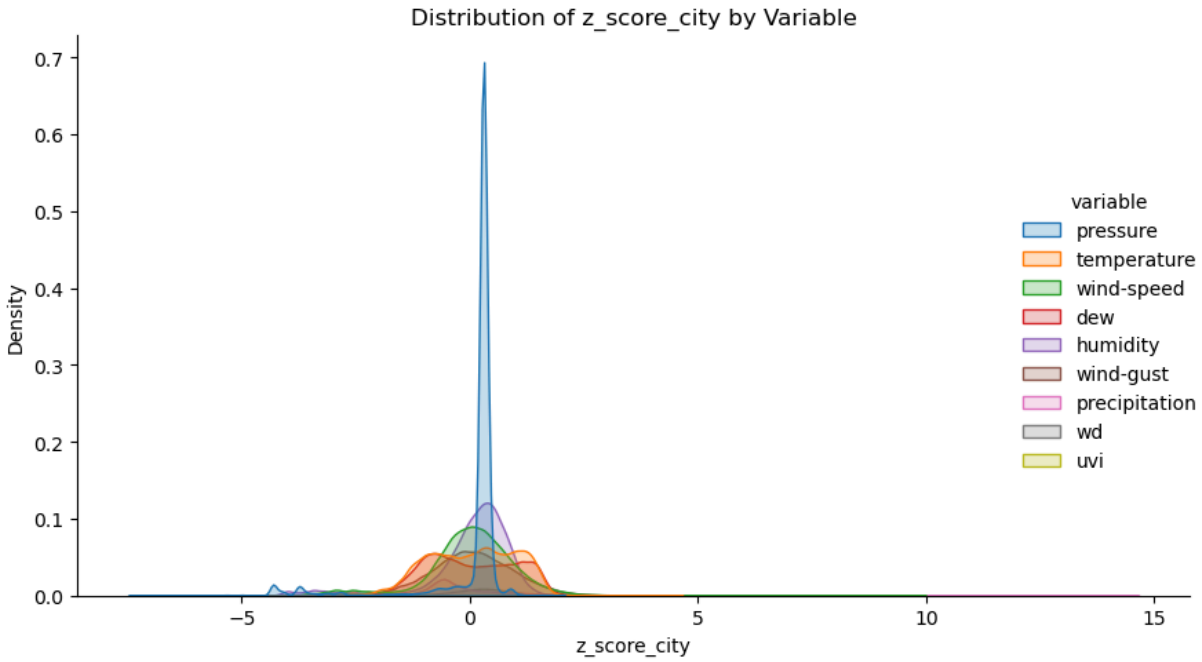


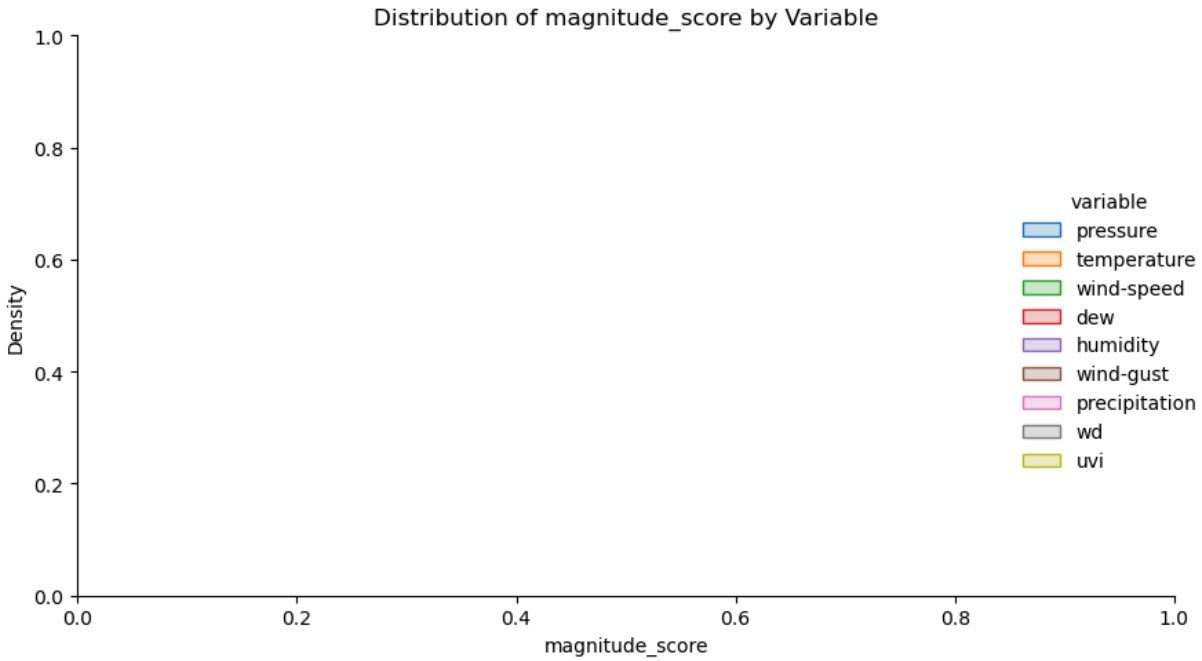
In [11]:
```
sql_file_path = os.path.join("..", "..", "03_SQL_queries", "01_data_exploration", "

with open(sql_file_path, 'r') as file:
    query = file.read()

df = pd.read_sql(query, engine)


print(df.head())
print(df.info())
```

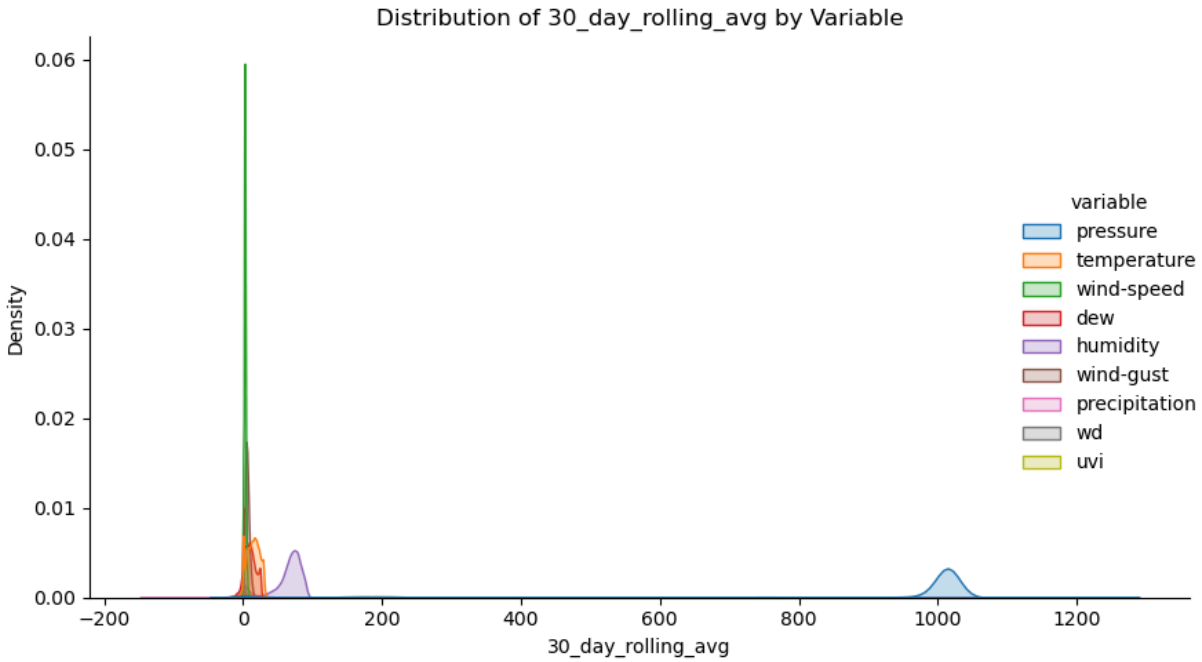
```
         date     city  variable     min     max  median  30_day_rolling_avg  \
0  2019-01-30  Quilpué  pressure  1013.0  1016.0  1015.0             1014.22
1  2019-01-01  Quilpué  pressure  1010.0  1013.0  1011.0             1011.00
2  2019-01-02  Quilpué  pressure  1008.0  1013.0  1011.0             1011.00
3  2019-01-03  Quilpué  pressure  1008.0  1011.0  1009.0             1010.33
4  2019-01-04  Quilpué  pressure  1009.0  1013.0  1012.0             1010.75

   z_score_all  z_score_city magnitude_score
0         0.34          0.27            None
1         0.33          0.26            None
2         0.33          0.26            None
3         0.33          0.25            None
4         0.33          0.26            None
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4999542 entries, 0 to 4999541
Data columns (total 10 columns):
 #   Column              Dtype
---  ------              -----
 0   date                object
 1   city                object
 2   variable            object
 3   min                 float64
 4   max                 float64
 5   median              float64
 6   30_day_rolling_avg  float64
 7   z_score_all         float64
 8   z_score_city        float64
 9   magnitude_score     object
dtypes: float64(6), object(4)
memory usage: 381.4+ MB
None
```

In [12]:
```python
for col in cal_cols:

    sns.displot(
        data=df,
        x=f"{col}",
        hue="variable",
        kind="kde", # use KDE to get a clearer view of the overall distribution
        fill=True,  # fills under the KDE curve for easier readability
        aspect=1.5
    )
    plt.title(f"Distribution of {col} by Variable")
    plt.tight_layout()
    plt.show()
```


Distribution of z_score_all by Variable
Density
z_score_all
variable
pressure
temperature
wind-speed
dew
humidity
wind-gust
precipitation
wd
uvi


Distribution of z_score_city by Variable
Density
z_score_city
variable
pressure
temperature
wind-speed
dew
humidity
wind-gust
precipitation
wd
uvi





In [14]:
```
notebook_path = os.path.join("..", "..", "03_python", "02_jupyter_notebooks", "data
output_dir = Path(notebook_path + "_charts")
output_dir.mkdir(exist_ok=True)

# Load the notebook content
with open(notebook_path, "r", encoding="utf-8") as f:
    nb = nbformat.read(f, as_version=4)

# Iterate over each cell in the notebook
for i, cell in enumerate(nb.cells):
    # Check if the cell has outputs and contains images
    if 'outputs' in cell:
        # Filter out cells with no chart (image/png)
        image_outputs = [output for output in cell['outputs'] if 'data' in output a
```

```
        # If there are no image outputs, skip the cell
        if not image_outputs:
            continue

        # Create a folder for the cell based on its index
        cell_folder = output_dir / f"cell_{i+1}"
        cell_folder.mkdir(exist_ok=True)

        image_counter = 1

        # Process each image output
        for output in image_outputs:
            # Decode the base64 image data
            image_data = output['data']['image/png']
            image_bytes = base64.b64decode(image_data)

            # Save image with a unique name based on the image count in the current
            image_filename = cell_folder / f"image_{image_counter}.png"
            with open(image_filename, "wb") as img_file:
                img_file.write(image_bytes)

            image_counter += 1

print(f"Images saved in '{output_dir}'")
```

```
Images saved in 'C:\Users\13476\Documents\GitHub\Air_Quality_Report_2019-2023\03_pyt
hon\02_jupyter_notebooks\40_aqi_full_report_data_exploration_2.ipynb_charts'
```

In [ ]: