In [ ]:

```
import requests
import pandas as pd
url = "https://data.cityofnewyork.us/resource/usep-8jbt.json"

params = {
    "$select": "borough, neighborhood, building_class_category, "
               "land_square_feet, gross_square_feet, sale_price, sale_date",
    "$where": "sale_price > 100000 AND "
              "gross_square_feet IS NOT NULL AND "
              "land_square_feet IS NOT NULL",
    "$limit": 1000
}

resp = requests.get(url, params=params)
df = pd.DataFrame(resp.json())
df
df.to_csv("/Users/lihuixiong/Desktop/DSCI510/FINAL/raw_data.csv", index=False)
```

In [4]:

```
numeric_cols = ["sale_price", "land_square_feet", "gross_square_feet"]
for col in numeric_cols:
    df[col] = pd.to_numeric(df[col], errors="coerce")

df["sale_date"] = pd.to_datetime(df["sale_date"], errors="coerce")


df = df.dropna(subset=["sale_price"])
df
```

Out[4]:

	borough	neighborhood	building_class_category	land_square_feet	gross_square_feet	sale_p
0	1	HARLEM-UPPER	11 SPECIAL CONDO BILLING LOTS	0.0	14659	37
1	1	MIDTOWN EAST	11 SPECIAL CONDO BILLING LOTS	0.0	2446	389
2	1	MIDTOWN EAST	11 SPECIAL CONDO BILLING LOTS	0.0	4321	850
3	3	DOWNTOWN-FULTON MALL	11 SPECIAL CONDO BILLING LOTS	0.0	129410	11280
4	3	WILLIAMSBURG-SOUTH	11 SPECIAL CONDO BILLING LOTS	0.0	89603	7260
...	...	...	...	...	...	
995	3	PARK SLOPE	02 TWO FAMILY DWELLINGS	NaN	3800	250
996	3	PARK SLOPE	02 TWO FAMILY DWELLINGS	NaN	3211	211
997	3	WILLIAMSBURG-NORTH	07 RENTALS - WALKUP APARTMENTS	NaN	3218	435
998	3	WILLIAMSBURG-SOUTH	02 TWO FAMILY DWELLINGS	NaN	2730	309
999	5	ARDEN HEIGHTS	01 ONE FAMILY DWELLINGS	NaN	1319	68

1000 rows × 7 columns

In [5]:

```
df.isna().sum()
```

Out[5]:

```
borough                      0
neighborhood                 0
building_class_category      0
land_square_feet           993
gross_square_feet            0
sale_price                   0
sale_date                    0
dtype: int64
```

In [6]:

```
value_counts = df["borough"].value_counts()


for cat, count in value_counts.items():
    print(f"{cat}: {count}")
```

```
3: 290
5: 258
4: 214
2: 140
1: 98
```

In [7]:

```
value_counts = df["building_class_category"].value_counts()


for cat, count in value_counts.items():
    print(f"{cat}: {count}")
```

```
01 ONE FAMILY DWELLINGS: 392
02 TWO FAMILY DWELLINGS: 197
22 STORE BUILDINGS: 57
07 RENTALS - WALKUP APARTMENTS: 53
03 THREE FAMILY DWELLINGS: 52
08 RENTALS - ELEVATOR APARTMENTS: 40
05 TAX CLASS 1 VACANT LAND: 40
29 COMMERCIAL GARAGES: 39
21 OFFICE BUILDINGS: 27
31 COMMERCIAL VACANT LAND: 23
14 RENTALS - 4-10 UNIT: 19
30 WAREHOUSES: 19
37 RELIGIOUS FACILITIES: 8
27 FACTORIES: 6
41 TAX CLASS 4 - OTHER: 5
32 HOSPITAL AND HEALTH FACILITIES: 5
11 SPECIAL CONDO BILLING LOTS: 5
06 TAX CLASS 1 - OTHER: 4
26 OTHER HOTELS: 4
33 EDUCATIONAL FACILITIES: 2
35 INDOOR PUBLIC AND CULTURAL FACILITIES: 2
38 ASYLUMS AND HOMES: 1
```

In [9]:

```python
def building_class_8cat(cat):
    if cat is None:
        return "Other"
    c = str(cat).upper()


    if c.startswith("01"):
        return "OneFamily"
    elif c.startswith("02"):
        return "TwoFamily"
    elif c.startswith("03"):
        return "ThreeFamily"


    elif c.startswith("07"):
        return "WalkupApt"
    elif c.startswith("08"):
        return "ElevatorApt"


    elif c.startswith("21") or c.startswith("22") or c.startswith("29") or c.starts
          or c.startswith("27") or c.startswith("33") or c.startswith("35") or c.sta
          or c.startswith("26"):
        return "Commercial"


    elif c.startswith("05") or c.startswith("31") or c.startswith("06"):
        return "VacantLand"


    else:
        return "Other"

df["building_class_8"] = df["building_class_category"].apply(building_class_8cat)
print(df["building_class_8"].value_counts())
```

```
OneFamily      392
TwoFamily      197
Commercial     161
VacantLand      67
WalkupApt       53
ThreeFamily     52
ElevatorApt     40
Other           38
Name: building_class_8, dtype: int64
```

In [ ]:

```
df = df.drop(columns=["land_square_feet","neighborhood"], errors="ignore")
df = df[df["gross_square_feet"] != 0]
df
df.to_csv("/Users/lihuixiong/Desktop/DSCI510/FINAL/processed_data.csv", index=False)
```

Out[ ]:

	borough	building_class_category	gross_square_feet	sale_price	sale_date	building_class_8
0	1	11 SPECIAL CONDO BILLING LOTS	14659	375000	2025-02-10	Other
1	1	11 SPECIAL CONDO BILLING LOTS	2446	3890000	2025-01-23	Other
2	1	11 SPECIAL CONDO BILLING LOTS	4321	8500000	2025-01-31	Other
3	3	11 SPECIAL CONDO BILLING LOTS	129410	112800000	2024-11-14	Other
4	3	11 SPECIAL CONDO BILLING LOTS	89603	72600000	2025-07-24	Other
...	...	...	...	...	...	...
995	3	02 TWO FAMILY DWELLINGS	3800	2500000	2025-09-04	TwoFamily
996	3	02 TWO FAMILY DWELLINGS	3211	2115000	2025-07-18	TwoFamily
997	3	07 RENTALS - WALKUP APARTMENTS	3218	4350000	2025-03-17	WalkupApt
998	3	02 TWO FAMILY DWELLINGS	2730	3090000	2025-08-07	TwoFamily
999	5	01 ONE FAMILY DWELLINGS	1319	680000	2025-07-18	OneFamily

915 rows × 6 columns

In [ ]:

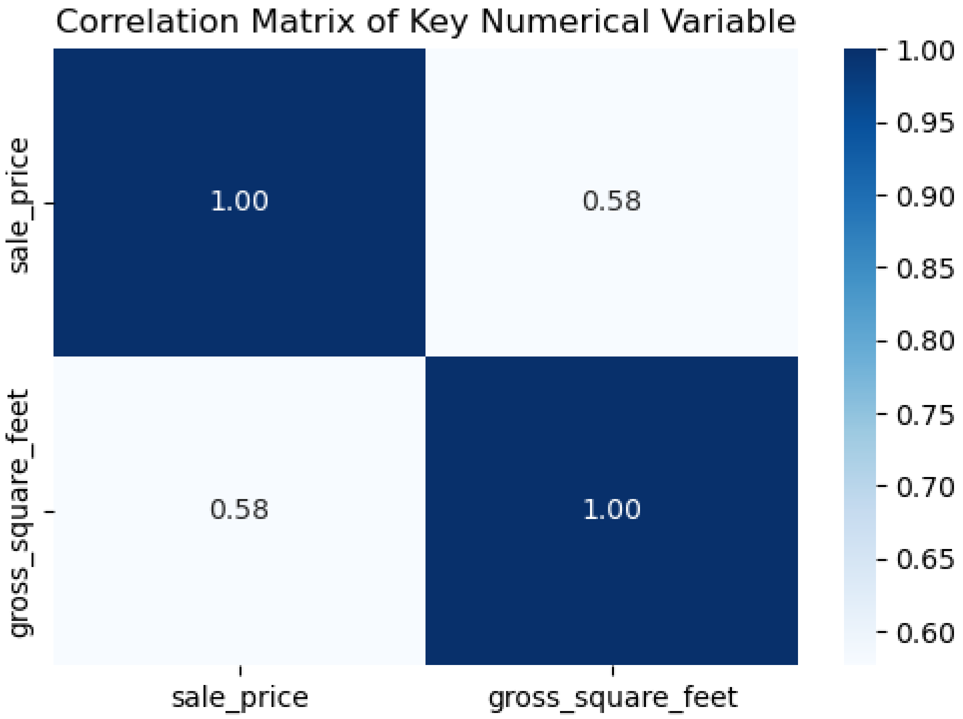
```
import numpy as np
df["log_price"] = np.log(df["sale_price"])
df["log_sqft"] = np.log(df["gross_square_feet"] + 1)

df[["sale_price", "gross_square_feet",
    "log_price", "log_sqft"]].describe()
```

Out[ ]:

	sale_price	gross_square_feet	log_price	log_sqft
count	9.150000e+02	915.000000	915.000000	915.000000
mean	4.383726e+06	9613.234973	14.302926	8.066457
std	1.235398e+07	23645.223555	1.158157	1.214271
min	1.020000e+05	420.000000	11.532728	6.042633
25%	6.500000e+05	1395.000000	13.384728	7.241364
50%	1.300000e+06	2301.000000	14.077875	7.741534
75%	3.345000e+06	4390.500000	15.022976	8.387426
max	1.640000e+08	238294.000000	18.915377	12.381265

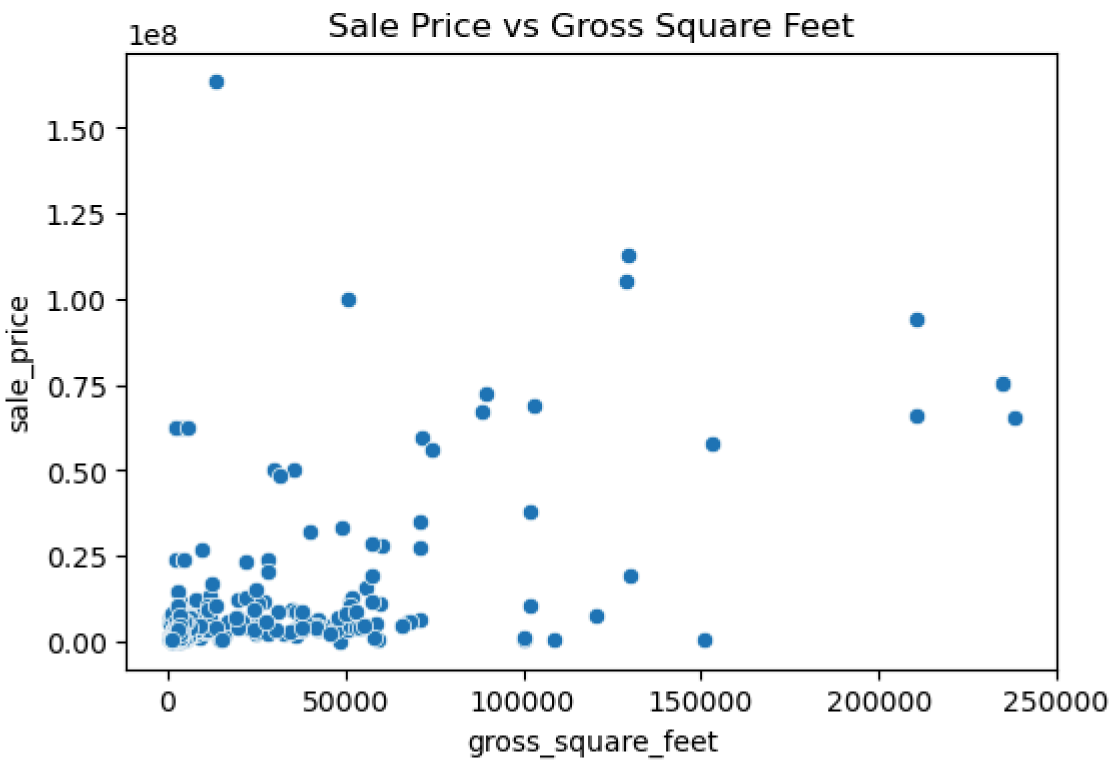
In [17]:

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

numeric_cols = ["sale_price", "gross_square_feet"]
corr = df[numeric_cols].corr()

plt.figure(figsize=(6,4))
sns.heatmap(corr, annot=True, cmap="Blues", fmt=".2f")
plt.title("Correlation Matrix of Key Numerical Variable")
plt.show()
```



In [19]:

```
plt.figure(figsize=(6,4))
sns.scatterplot(x=df["gross_square_feet"], y=df["sale_price"])
plt.title("Sale Price vs Gross Square Feet")
plt.show()
```



In [ ]:

```
from sklearn.preprocessing import OneHotEncoder, StandardScaler
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.linear_model import LinearRegression, Ridge, Lasso
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.metrics import r2_score
from sklearn.ensemble import RandomForestRegressor




numeric_features = ["log_sqft"]
categorical_features = ["building_class_8", "borough"]


numeric_transformer = Pipeline(steps=[
    ("scaler", StandardScaler())
])


categorical_transformer = Pipeline(steps=[
    ("onehot", OneHotEncoder(handle_unknown="ignore"))
])

# ColumnTransformer
preprocess = ColumnTransformer(
    transformers=[
        ("num", numeric_transformer, numeric_features),
        ("cat", categorical_transformer, categorical_features)
    ]
)


X = df[numeric_features + categorical_features]
y = df["log_price"]

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)
```

```
model_lr = Pipeline(steps=[
    ("preprocess", preprocess),
    ("lr", LinearRegression())
])

model_lr.fit(X_train, y_train)
lr_r2 = model_lr.score(X_test, y_test)
print("Linear Regression R²:", lr_r2)
```

```
Linear Regression R²: 0.4242983118986853
```

In [ ]:

```
df["sale_date"] = pd.to_datetime(df["sale_date"], errors="coerce")
df["year"] = df["sale_date"].dt.year
df["month"] = df["sale_date"].dt.month
df["month_sin"] = np.sin(2*np.pi*df["month"]/12)
df["month_cos"] = np.cos(2*np.pi*df["month"]/12)

df["log_price"] = np.log(df["sale_price"])
df["log_sqft"] = np.log(df["gross_square_feet"] + 1)

df["sqft_borough_interact"] = df["log_sqft"] * df["borough"].astype("category").cat.
df["sqft_class_interact"] = df["log_sqft"] * df["building_class_8"].astype("category

numeric_features = [
    "log_sqft",
    "year",
    "month_sin",
    "month_cos",
    "sqft_borough_interact",
    "sqft_class_interact"
]

categorical_features = ["borough", "building_class_8"]

numeric_transformer = Pipeline(steps=[
    ("scaler", StandardScaler())
])

categorical_transformer = Pipeline(steps=[
    ("onehot", OneHotEncoder(handle_unknown="ignore"))
])

preprocess = ColumnTransformer(
    transformers=[
        ("num", numeric_transformer, numeric_features),
        ("cat", categorical_transformer, categorical_features)
    ]
)

X = df[numeric_features + categorical_features]
y = df["log_price"]

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)


# Linear Regression
lr = Pipeline(steps=[
    ("preprocess", preprocess),
    ("lr", LinearRegression())
])
lr.fit(X_train, y_train)
print("LR R²:", lr.score(X_test, y_test))
```

```
LR R²: 0.4425669849105376
```

In [ ]:

```python
coef = lr.named_steps["lr"].coef_
intercept = lr.named_steps["lr"].intercept_

print("Intercept:", intercept)
```

```
Intercept: 14.488499333503762
```

In [ ]:

```python
ohe = lr.named_steps["preprocess"].named_transformers_["cat"]["onehot"]

ohe_feature_names = ohe.get_feature_names_out(categorical_features)


all_features = numeric_features + list(ohe_feature_names)

print("Total features:", len(all_features))
print(all_features)

coef_df = pd.DataFrame({
    "feature": all_features,
    "coefficient": coef
})
```

```
Total features: 18
['log_sqft', 'year', 'month_sin', 'month_cos', 'sqft_borough_interact', 'sqft_class_
interact', 'borough_1', 'borough_2', 'borough_3', 'borough_4', 'borough_5', 'buildin
g_class_8_Commercial', 'building_class_8_ElevatorApt', 'building_class_8_OneFamily',
'building_class_8_Other', 'building_class_8_ThreeFamily', 'building_class_8_TwoFamil
y', 'building_class_8_WalkupApt']
```

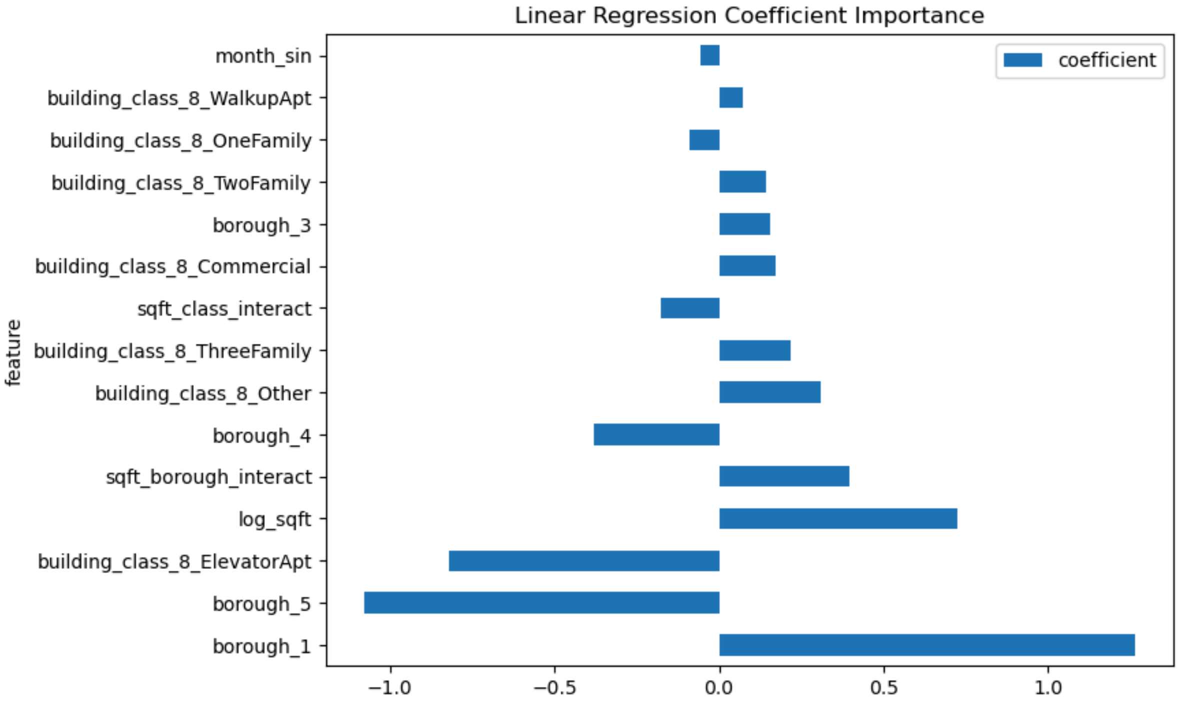
In [33]:

```python
coef_df["abs_coef"] = coef_df["coefficient"].abs()
coef_df_sorted = coef_df.sort_values("abs_coef", ascending=False)

print(coef_df_sorted)
```

```
                         feature  coefficient  abs_coef
6                      borough_1     1.264093  1.264093
10                     borough_5    -1.077126  1.077126
12  building_class_8_ElevatorApt    -0.822455  0.822455
0                       log_sqft     0.723789  0.723789
4          sqft_borough_interact     0.398530  0.398530
9                      borough_4    -0.378739  0.378739
14        building_class_8_Other     0.307773  0.307773
15  building_class_8_ThreeFamily     0.217013  0.217013
5            sqft_class_interact    -0.176694  0.176694
11   building_class_8_Commercial     0.173645  0.173645
8                      borough_3     0.153660  0.153660
16    building_class_8_TwoFamily     0.141824  0.141824
13    building_class_8_OneFamily    -0.089779  0.089779
17    building_class_8_WalkupApt     0.071980  0.071980
2                      month_sin    -0.056721  0.056721
7                      borough_2     0.038112  0.038112
1                           year     0.027334  0.027334
3                      month_cos    -0.013495  0.013495
```

In [ ]:

```python
import matplotlib.pyplot as plt

topN = 15
coef_df_sorted.head(topN).plot(
    x="feature", y="coefficient", kind="barh", figsize=(8,6)
)
plt.title("Linear Regression Coefficient Importance")
plt.show()
```


Linear Regression Coefficient Importance
coefficient
feature
month_sin
building_class_8_WalkupApt
building_class_8_OneFamily
building_class_8_TwoFamily
borough_3
building_class_8_Commercial
sqft_class_interact
building_class_8_ThreeFamily
building_class_8_Other
borough_4
sqft_borough_interact
log_sqft
building_class_8_ElevatorApt
borough_5
borough_1
−1.0
−0.5
0.0
0.5
1.0