In [26]:
```
# housing_regression.py
# A complete linear regression for the housing dataset
```

In [27]:
```
# 1. Import libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy import stats

from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score, mean_squared_error

import statsmodels.api as sm
from statsmodels.stats.outliers_influence import variance_inflation_fa
```

In [28]:
```
# 2. Load the dataset
df = pd.read_csv('housing.csv')
```

In [29]:
```
# 3. Initial inspection & cleaning
print("=== First five rows ===")
print(df.head(), "\n")

print("=== Data summary (numeric columns) ===")
print(df.describe().T, "\n")

print("=== Missing values per column ===")
print(df.isnull().sum(), "\n")
```

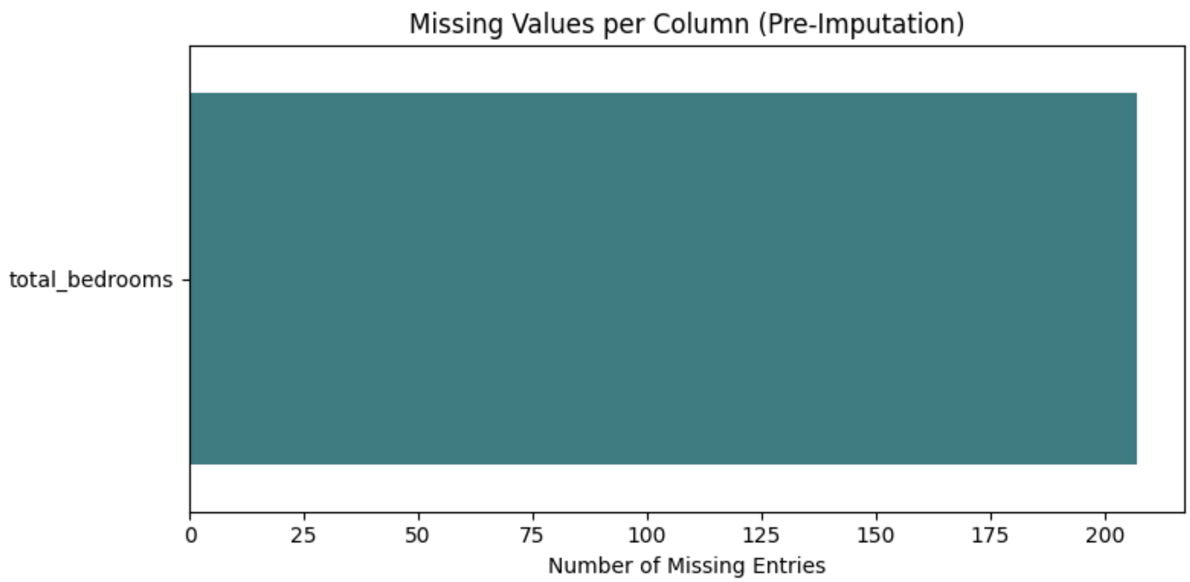
```
=== First five rows ===
   longitude  latitude  housing_median_age  total_rooms  total_bedrooms
\
0    -122.23     37.88                41.0        880.0           129.0
1    -122.22     37.86                21.0       7099.0          1106.0
2    -122.24     37.85                52.0       1467.0           190.0
3    -122.25     37.85                52.0       1274.0           235.0
4    -122.25     37.85                52.0       1627.0           280.0

   population  households  median_income  median_house_value ocean_prox
imity
0       322.0       126.0         8.3252            452600.0        NEA
R BAY
1      2401.0      1138.0         8.3014            358500.0        NEA
R BAY
2       496.0       177.0         7.2574            352100.0        NEA
R BAY
3       558.0       219.0         5.6431            341300.0        NEA
R BAY
4       565.0       259.0         3.8462            342200.0        NEA
```

```
R BAY

=== Data summary (numeric columns) ===
                      count           mean            std          min  \
longitude           20640.0    -119.569704       2.003532    -124.3500
latitude            20640.0      35.631861       2.135952      32.5400
housing_median_age  20640.0      28.639486      12.585558       1.0000
total_rooms         20640.0    2635.763081    2181.615252       2.0000
total_bedrooms      20433.0     537.870553     421.385070       1.0000
population          20640.0    1425.476744    1132.462122       3.0000
households          20640.0     499.539680     382.329753       1.0000
median_income       20640.0       3.870671       1.899822       0.4999
median_house_value  20640.0  206855.816909  115395.615874   14999.0000

                            25%          50%           75%          max
longitude             -121.8000    -118.4900    -118.01000    -114.3100
latitude                33.9300      34.2600      37.71000      41.9500
housing_median_age      18.0000      29.0000      37.00000      52.0000
total_rooms           1447.7500    2127.0000    3148.00000   39320.0000
total_bedrooms         296.0000     435.0000     647.00000    6445.0000
population             787.0000    1166.0000    1725.00000   35682.0000
households             280.0000     409.0000     605.00000    6082.0000
median_income            2.5634       3.5348       4.74325      15.0001
median_house_value  119600.0000  179700.0000  264725.00000  500001.0000

=== Missing values per column ===
longitude               0
latitude                0
housing_median_age      0
total_rooms             0
total_bedrooms        207
population              0
households              0
median_income           0
median_house_value      0
ocean_proximity         0
dtype: int64
```

In [30]:
```
# Figure 1 – Barplot of missing values before imputation
# ---------------------------------------------------------
# Run this *before* the median fill step.

import matplotlib.pyplot as plt
import seaborn as sns

# Count NaNs column-wise
na_counts = df.isnull().sum()
na_counts = na_counts[na_counts > 0].sort_values()

plt.figure(figsize=(8, 4))
sns.barplot(
```

```
    x=na_counts.values,
    y=na_counts.index,
    palette="crest",
    orient="h"
)
plt.title("Missing Values per Column (Pre-Imputation)")
plt.xlabel("Number of Missing Entries")
plt.ylabel("")
plt.tight_layout()
plt.show()
```

```
/var/folders/nk/ll9_xqj958jb6d031j0rwjvw0000gn/T/ipykernel_62453/378498
453.py:13: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be rem
oved in v0.14.0. Assign the `y` variable to `hue` and set `legend=False
` for the same effect.

  sns.barplot(
```

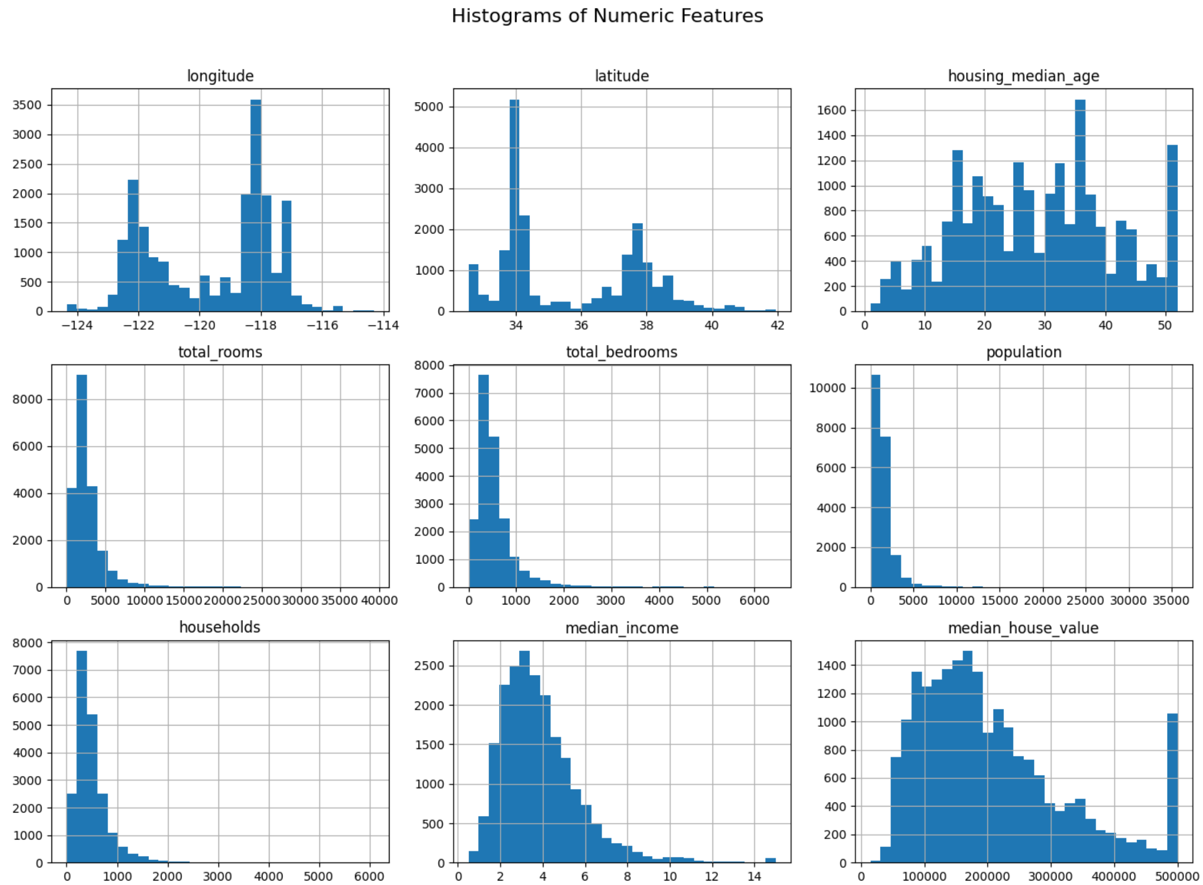



In [31]:
```
# 3.1 Categorical encoding – one-hot for ocean_proximity (required for
if "ocean_proximity" in df.columns:
    df = pd.get_dummies(df, columns=["ocean_proximity"], drop_first=Tr
else:
    print("Note: 'ocean_proximity' already one-hot encoded – skipping 

missing_total = df.isnull().sum().sum()
if missing_total > 0:
    num_cols = df.select_dtypes(include=[np.number]).columns
    df[num_cols] = df[num_cols].fillna(df[num_cols].median())
    print(f"Imputed {missing_total} missing numeric values with column
else:
    print("No missing numeric values to impute.")
```

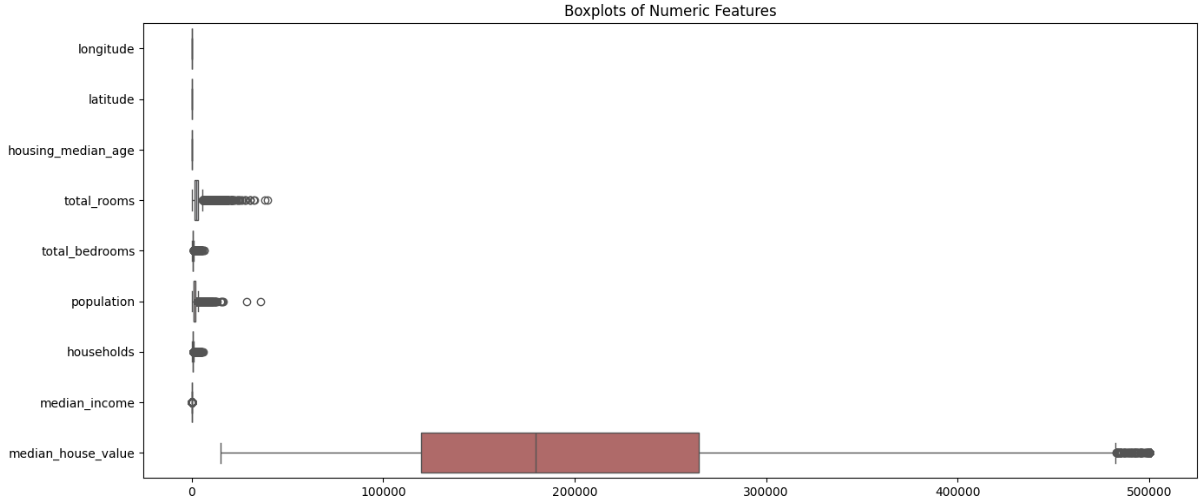
```
Imputed 207 missing numeric values with column medians.
```

In [32]:
```
# 4. Exploratory Data Analysis (EDA) ─────────────────────────────
numeric_cols = df.select_dtypes(include=[np.number]).columns
```

In [33]:
```
# 4.1 Histograms
_ = df[numeric_cols].hist(figsize=(14, 10), bins=30)
plt.suptitle("Histograms of Numeric Features", y=1.02, fontsize=16)
plt.tight_layout()
plt.show()
```

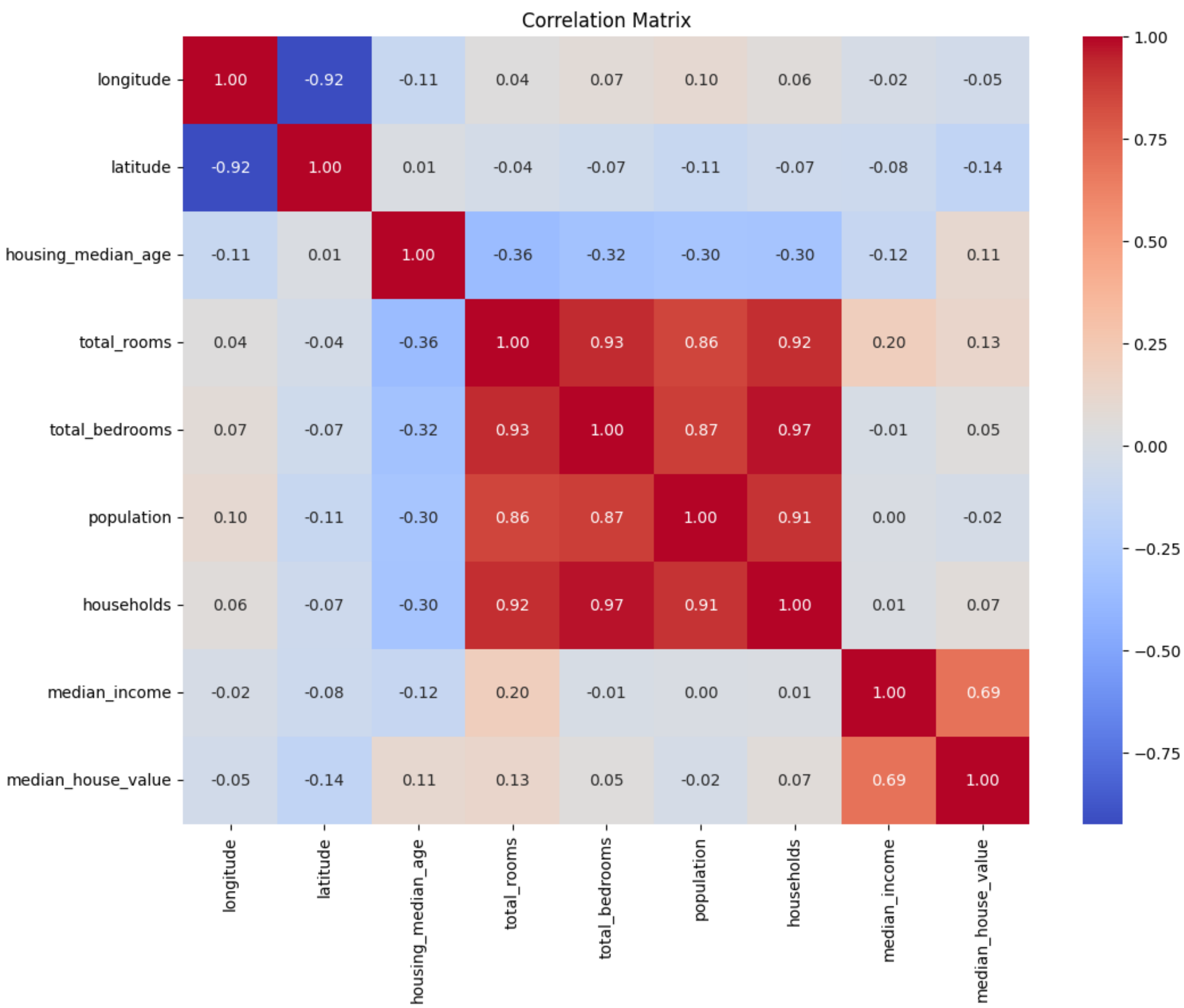



In [34]:
```
# 4.2 Boxplots
plt.figure(figsize=(14, 6))
sns.boxplot(data=df[numeric_cols], orient="h", palette="vlag")
plt.title("Boxplots of Numeric Features")
plt.tight_layout()
plt.show()
```




In [35]:

```python
# 4.3 Correlation heatmap
plt.figure(figsize=(12, 9))
corr = df[numeric_cols].corr()
sns.heatmap(corr, annot=True, cmap="coolwarm", fmt=".2f")
plt.title("Correlation Matrix")
plt.show()
```




In [36]:

```python
# 5. Outlier detection (report only, no removal) ──────────────────
```

```
Z = np.abs(stats.zscore(df[numeric_cols]))
outlier_mask = (Z >= 3).any(axis=1)
print(f"Rows with |Z| ≥ 3: {outlier_mask.sum()} (reported, not removed
```

```
Rows with |Z| ≥ 3: 894 (reported, not removed)
```

In [37]:
```
# 6. Train-test split ──────────────────────────────────────
TARGET = "median_house_value"
X = df.drop(TARGET, axis=1)
y = df[TARGET]

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.20, random_state=42
)
```

In [38]:
```
#regression equation

coef_series = pd.Series(linreg.coef_, index=X.columns)
eqn = "median_house_value = " + " + ".join(
    f"{b:.4f}*{c}" for c, b in coef_series.items()
) + f" + {linreg.intercept_:.4f}"
print("\nRegression equation:\n", eqn)
```

```
Regression equation:
 median_house_value = -26838.2734*longitude + -25468.3520*latitude + 11
02.1851*housing_median_age + -6.0215*total_rooms + 102.7894*total_bedro
oms + -38.1729*population + 48.2528*households + 39473.9752*median_inco
me + -39786.6562*ocean_proximity_INLAND + 136125.0726*ocean_proximity_I
SLAND + -5136.6422*ocean_proximity_NEAR BAY + 3431.1401*ocean_proximity
_NEAR OCEAN + -2275547.3817
```

In [39]:
```
# 7.1 scikit-learn fit
linreg = LinearRegression()
linreg.fit(X_train, y_train)
y_pred = linreg.predict(X_test)

# 7.2 statsmodels OLS for detailed stats
X_train_sm = sm.add_constant(X_train).astype(float)
# Ensure y is float
y_train_float = y_train.astype(float)
ols_model = sm.OLS(y_train_float, X_train_sm).fit()
print(ols_model.summary())
```

```
                            OLS Regression Results
=======================================================================
=======
Dep. Variable:     median_house_value   R-squared:
0.650
Model:                            OLS   Adj. R-squared:
0.649
Method:                 Least Squares   F-statistic:
2550.
```

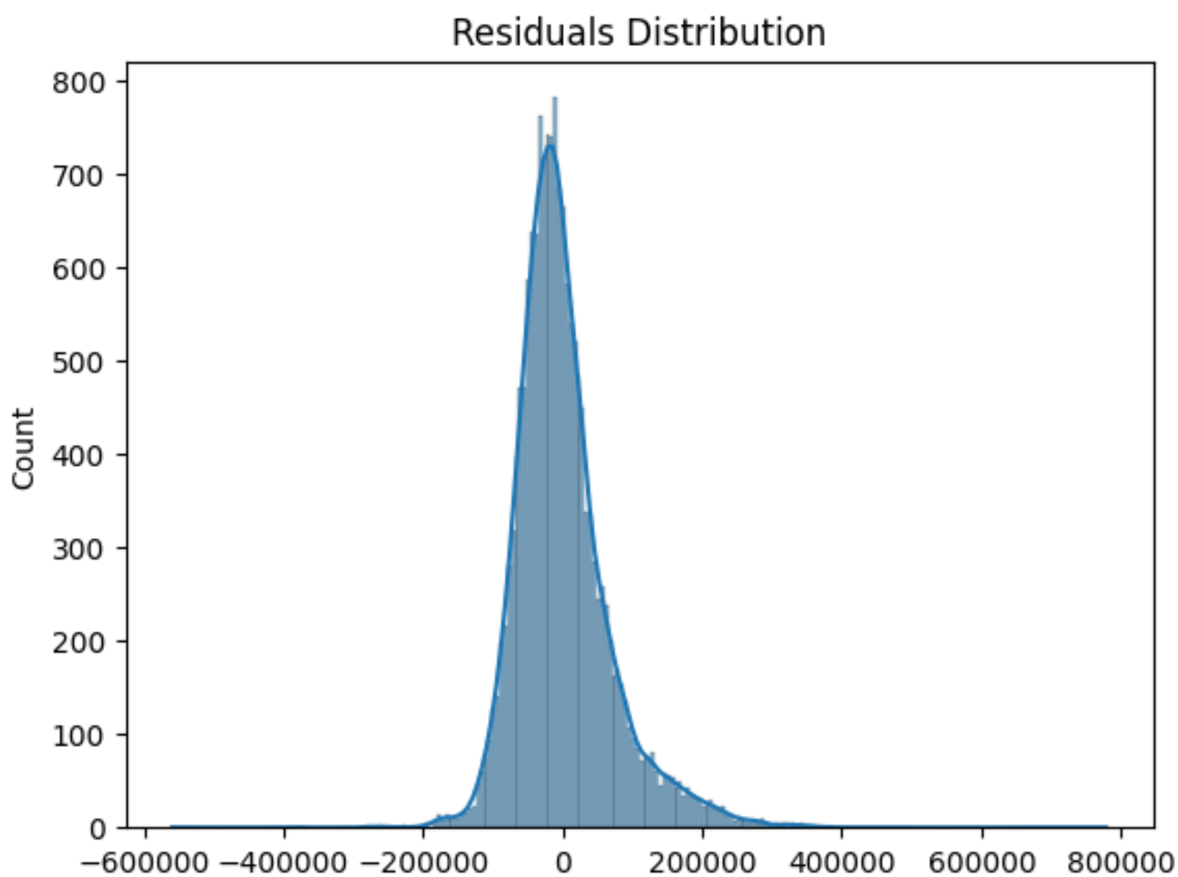
```
Date:                  Thu, 24 Apr 2025   Prob (F-statistic):
0.00
Time:                          12:01:43   Log-Likelihood:            -2.0
727e+05
No. Observations:                 16512   AIC:                         4.
146e+05
Df Residuals:                     16499   BIC:                         4.
147e+05
Df Model:                            12
Covariance Type:              nonrobust
==============================================================================
=======================
                                coef    std err          t      P>|t|
[0.025      0.975]
------------------------------------------------------------------------------
-----------------------
const                     -2.276e+06   9.73e+04    -23.394      0.000
-2.47e+06   -2.08e+06
longitude                 -2.684e+04   1127.047    -23.813      0.000
-2.9e+04   -2.46e+04
latitude                  -2.547e+04   1111.486    -22.914      0.000
-2.76e+04   -2.33e+04
housing_median_age         1102.1851     48.605     22.676      0.000
1006.914    1197.456
total_rooms                  -6.0215      0.886     -6.796      0.000
-7.758      -4.285
total_bedrooms              102.7894      7.697     13.355      0.000
87.703     117.876
population                  -38.1729      1.188    -32.129      0.000
-40.502     -35.844
households                   48.2528      8.375      5.761      0.000
31.836      64.669
median_income              3.947e+04    375.091    105.238      0.000
3.87e+04    4.02e+04
ocean_proximity_INLAND    -3.979e+04   1933.681    -20.576      0.000
-4.36e+04    -3.6e+04
ocean_proximity_ISLAND     1.361e+05   3.43e+04      3.972      0.000
6.89e+04    2.03e+05
ocean_proximity_NEAR BAY  -5136.6422   2111.676     -2.432      0.015
-9275.756    -997.529
ocean_proximity_NEAR OCEAN  3431.1401   1751.612      1.959      0.050
-2.208    6864.488
==============================================================================
=======
Omnibus:                     4119.707   Durbin-Watson:
1.967
Prob(Omnibus):                  0.000   Jarque-Bera (JB):            16
516.873
Skew:                           1.189   Prob(JB):
0.00
Kurtosis:                       7.284   Cond. No.
7.21e+05
```

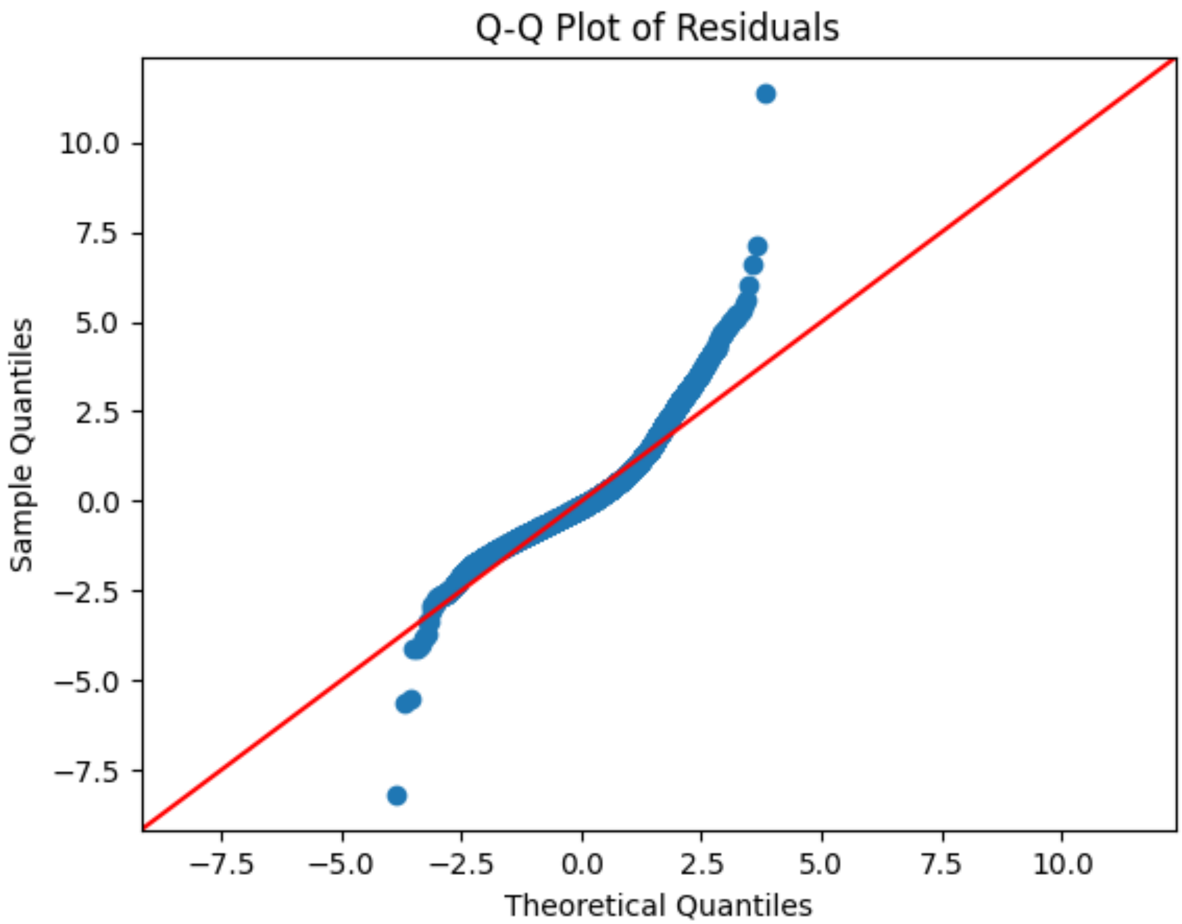
```
==============================================================================
=======

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is
correctly specified.
[2] The condition number is large, 7.21e+05. This might indicate that t
here are
strong multicollinearity or other numerical problems.
```

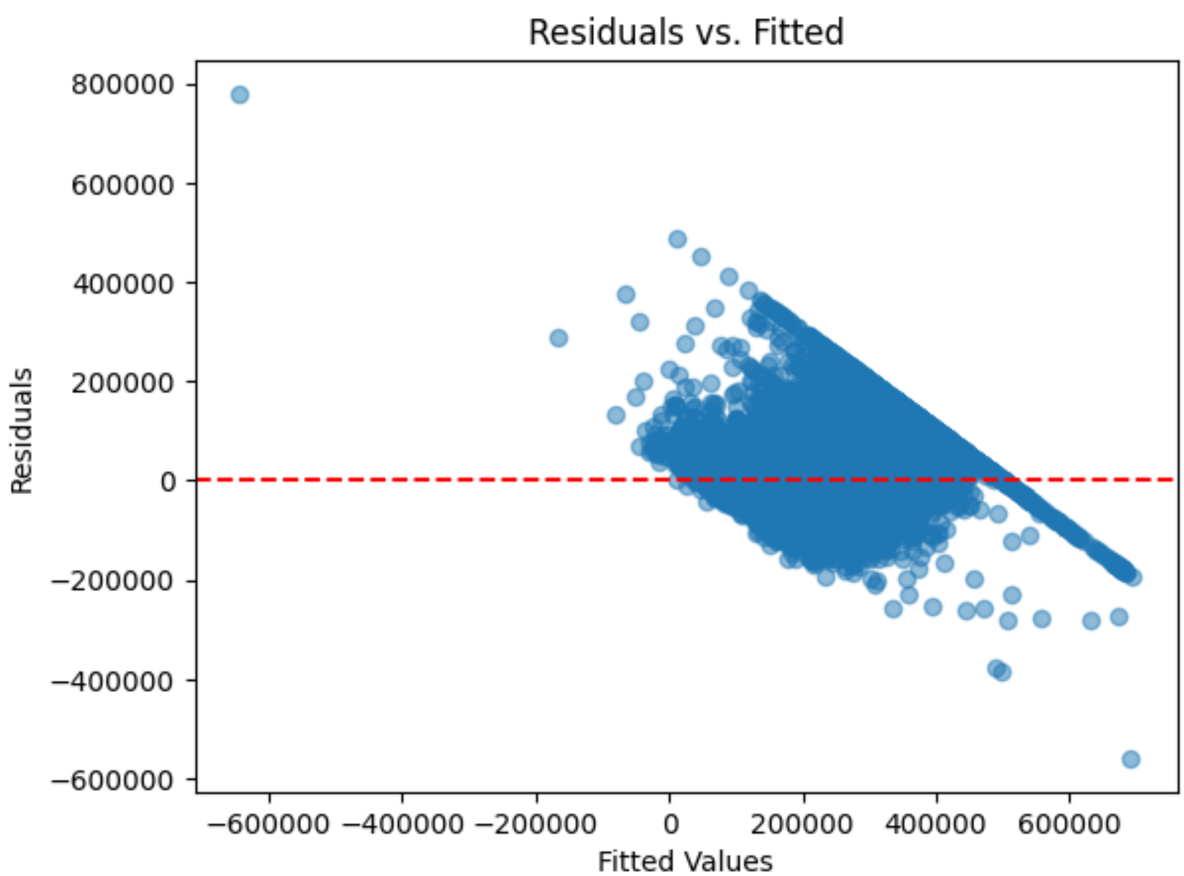
In [40]:

```python
# 8. Diagnostics ──────────────────────────────────────────────────────────
# 8.1 VIF
vif = pd.DataFrame({
    "feature": X_train_sm.columns,
    "VIF": [variance_inflation_factor(X_train_sm.values, i) for i in r
})
print("\n=== Variance Inflation Factors ===")
print(vif, "\n")

# 8.2 Residual analysis
residuals = y_train_float - ols_model.predict(X_train_sm)

# Histogram
sns.histplot(residuals, kde=True)
plt.title("Residuals Distribution")
plt.show()

# Q-Q plot
sm.qqplot(residuals, line="45", fit=True)
plt.title("Q-Q Plot of Residuals")
plt.show()

# Residuals vs fitted
plt.scatter(ols_model.predict(X_train_sm), residuals, alpha=0.5)
plt.axhline(0, color="red", ls="--")
plt.xlabel("Fitted Values")
plt.ylabel("Residuals")
plt.title("Residuals vs. Fitted")
plt.show()

# Normality & autocorrelation tests
sh_w, sh_p = stats.shapiro(residuals)
dw = sm.stats.stattools.durbin_watson(residuals)
print(f"Shapiro-Wilk: W = {sh_w:.3f}, p = {sh_p:.4f}")
print(f"Durbin-Watson : {dw:.3f}\n")
```

```
=== Variance Inflation Factors ===
                        feature           VIF
0                         const  33333.996165
1                     longitude     18.000488
2                      latitude     19.868698
3            housing_median_age      1.321783
4                   total_rooms     13.078025
5                total_bedrooms     36.638126
6                    population      6.429272
7                    households     35.864528
8                 median_income      1.797340
9        ocean_proximity_INLAND      2.849961
10       ocean_proximity_ISLAND      1.002189
11     ocean_proximity_NEAR BAY      1.565865
12   ocean_proximity_NEAR OCEAN      1.193029
```




Q-Q Plot of Residuals
Sample Quantiles
Theoretical Quantiles
10.0
7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5


Residuals vs. Fitted
Residuals
Fitted Values
800000
600000
400000
200000
0
−200000
−400000
−600000

```
Shapiro-Wilk: W = 0.926, p = 0.0000
Durbin-Watson : 1.967
```

```
/Users/krishilparmar/Desktop/SP JAIN/Semester_2/Statistics/Project/ven
v/lib/python3.13/site-packages/scipy/stats/_axis_nan_policy.py:586: Use
rWarning: scipy.stats.shapiro: For N > 5000, computed p-value may not b
e accurate. Current N is 16512.
  res = hypotest_fun_out(*samples, **kwds)
```

In [41]:

```python
import statsmodels.formula.api as smf
from statsmodels.stats.anova import anova_lm

# --- 1.  Make column names safe for patsy (no spaces or punctuation)
safe_cols = {c: c.replace(" ", "_").replace(">", "gt") for c in df.col
df_ren = df.rename(columns=safe_cols)

# --- 2.  Build formula string ---
TARGET = "median_house_value"
predictors = [c for c in df_ren.columns if c != TARGET]
formula = TARGET + " ~ " + " + ".join(predictors)

# --- 3.  Fit formula-based OLS on *training* rows only (to keep parit
train_idx = X_train.index                        # same rows used earlier
ols_formula = smf.ols(formula, data=df_ren.loc[train_idx]).fit()

# --- 4.  Print summary (optional) ---
print(ols_formula.summary())

# --- 5.  Type-II ANOVA table (matches sample report) ---
anova_tbl = anova_lm(ols_formula, typ=2)
print("\n=== ANOVA Table (Type II) ===\n", anova_tbl)
```

```
                            OLS Regression Results
==============================================================================
=======
Dep. Variable:     median_house_value   R-squared:
0.650
Model:                            OLS   Adj. R-squared:
0.649
Method:                 Least Squares   F-statistic:
2550.
Date:                Thu, 24 Apr 2025   Prob (F-statistic):
0.00
Time:                        12:01:43   Log-Likelihood:            -2.0
727e+05
No. Observations:               16512   AIC:                         4.
146e+05
Df Residuals:                   16499   BIC:                         4.
147e+05
Df Model:                          12
Covariance Type:            nonrobust
==============================================================================
===============================
```

```
                                          coef    std err          t
P>|t|      [0.025      0.975]
------------------------------------------------------------------------
-------------------------------
Intercept                          -2.276e+06   9.73e+04    -23.394
0.000   -2.47e+06   -2.08e+06
ocean_proximity_INLAND[T.True]     -3.979e+04   1933.681    -20.576
0.000   -4.36e+04    -3.6e+04
ocean_proximity_ISLAND[T.True]      1.361e+05   3.43e+04      3.972
0.000    6.89e+04    2.03e+05
ocean_proximity_NEAR_BAY[T.True]   -5136.6422   2111.676     -2.432
0.015   -9275.756    -997.529
ocean_proximity_NEAR_OCEAN[T.True]  3431.1401   1751.612      1.959
0.050      -2.208    6864.488
longitude                          -2.684e+04   1127.047    -23.813
0.000    -2.9e+04   -2.46e+04
latitude                           -2.547e+04   1111.486    -22.914
0.000   -2.76e+04   -2.33e+04
housing_median_age                  1102.1851     48.605     22.676
0.000    1006.914    1197.456
total_rooms                           -6.0215      0.886     -6.796
0.000      -7.758      -4.285
total_bedrooms                       102.7894      7.697     13.355
0.000      87.703     117.876
population                           -38.1729      1.188    -32.129
0.000     -40.502     -35.844
households                            48.2528      8.375      5.761
0.000      31.836      64.669
median_income                       3.947e+04    375.091    105.238
0.000    3.87e+04    4.02e+04
========================================================================
=======
Omnibus:                     4119.707   Durbin-Watson:
1.967
Prob(Omnibus):                  0.000   Jarque-Bera (JB):            16
516.873
Skew:                           1.189   Prob(JB):
0.00
Kurtosis:                       7.284   Cond. No.
7.21e+05
========================================================================
=======

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is
correctly specified.
[2] The condition number is large, 7.21e+05. This might indicate that t
here are
strong multicollinearity or other numerical problems.

=== ANOVA Table (Type II) ===
                                   sum_sq       df             F
```

```
PR(>F)
ocean_proximity_INLAND      1.984222e+12      1.0     423.355478   6.641
048e-93
ocean_proximity_ISLAND      7.394035e+10      1.0      15.775981   7.160
469e-05
ocean_proximity_NEAR_BAY    2.773250e+10      1.0       5.917031   1.500
576e-02
ocean_proximity_NEAR_OCEAN  1.798400e+10      1.0       3.837083   5.014
747e-02
longitude                   2.657725e+12      1.0     567.054691  2.9246
39e-123
latitude                    2.460813e+12      1.0     525.041378  2.0559
30e-114
housing_median_age          2.410103e+12      1.0     514.221708  3.9364
07e-112
total_rooms                 2.164805e+11      1.0      46.188486   1.110
821e-11
total_bedrooms              8.359485e+11      1.0     178.358736   1.794
557e-40
population                  4.838225e+12      1.0    1032.288160  9.6000
23e-220
households                  1.555713e+11      1.0      33.192828   8.494
516e-09
median_income               5.190783e+13      1.0   11075.101861   0.000
000e+00
Residual                    7.732906e+13  16499.0            NaN
NaN
```

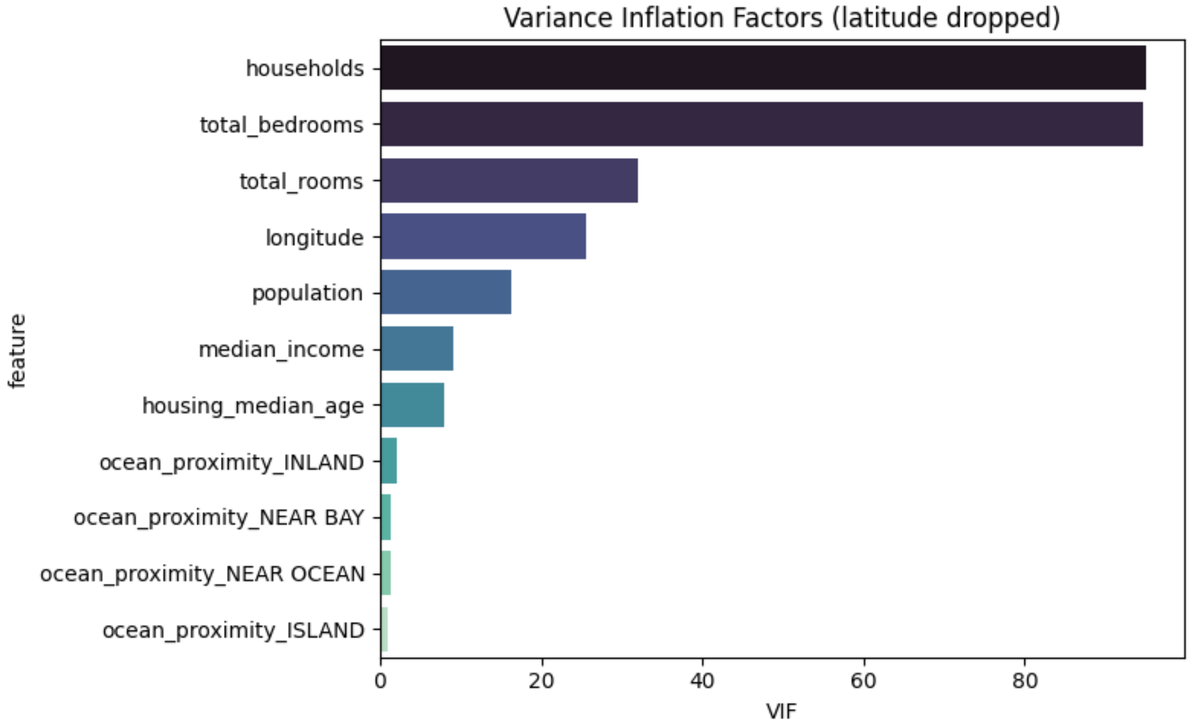
In [42]:
```python
# --- Type-II ANOVA on the training data -----------------------------
import statsmodels.formula.api as smf
from statsmodels.stats.anova import anova_lm

# 1) Make column names patsy-safe (replace spaces, symbols)
safe_map = {c: c.replace(" ", "_").replace("<", "lt").replace(">", "gt
            for c in df.columns}
df_safe   = df.rename(columns=safe_map)

# 2) Build formula string:  median_house_value ~ all other columns
TARGET = "median_house_value"
predictors = [c for c in df_safe.columns if c != TARGET]
formula = TARGET + " ~ " + " + ".join(predictors)

# 3) Fit formula OLS **on training rows only**
train_idx   = X_train.index              # reuse earlier split
ols_formula = smf.ols(formula, data=df_safe.loc[train_idx]).fit()

# 4) Type-II sums of squares
anova_tbl = anova_lm(ols_formula, typ=2)   # columns: DF, sum_sq, F, P
print("\n=== Type-II ANOVA Table ===\n")
print(anova_tbl.round(2))                  # <-- Table 4 for the repor
```

```
=== Type-II ANOVA Table ===

                                  sum_sq       df         F  PR(>F)
ocean_proximity_INLAND      1.984222e+12      1.0    423.36    0.00
ocean_proximity_ISLAND      7.394035e+10      1.0     15.78    0.00
ocean_proximity_NEAR_BAY    2.773250e+10      1.0      5.92    0.02
ocean_proximity_NEAR_OCEAN  1.798400e+10      1.0      3.84    0.05
longitude                   2.657725e+12      1.0    567.05    0.00
latitude                    2.460813e+12      1.0    525.04    0.00
housing_median_age          2.410103e+12      1.0    514.22    0.00
total_rooms                 2.164805e+11      1.0     46.19    0.00
total_bedrooms              8.359485e+11      1.0    178.36    0.00
population                  4.838225e+12      1.0   1032.29    0.00
households                  1.555713e+11      1.0     33.19    0.00
median_income               5.190783e+13      1.0  11075.10    0.00
Residual                    7.732906e+13  16499.0       NaN     NaN
```

In [43]:

```
# ------------------------------------------
#  A)   Drop 'latitude' before VIF analysis
# ------------------------------------------
X_vif = X_train.drop(columns=['latitude'])   # keep longitude as proxy

# Cast to float
X_vif = X_vif.astype(float)

vif_df = pd.DataFrame({
    'feature': X_vif.columns,
    'VIF': [variance_inflation_factor(X_vif.values, i)
            for i in range(X_vif.shape[1])]
})

# Plot
plt.figure(figsize=(8, 5))
sns.barplot(data=vif_df.sort_values('VIF', ascending=False),
            x='VIF', y='feature', palette='mako')
plt.title('Variance Inflation Factors (latitude dropped)')
plt.tight_layout()
plt.show()

print(vif_df.sort_values('VIF', ascending=False))
```

```
/var/folders/nk/ll9_xqj958jb6d031j0rwjvw0000gn/T/ipykernel_62453/112687
1815.py:17: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be rem
oved in v0.14.0. Assign the `y` variable to `hue` and set `legend=False
` for the same effect.

  sns.barplot(data=vif_df.sort_values('VIF', ascending=False),
```




```
                       feature        VIF
5                   households  95.046604
3               total_bedrooms  94.770052
2                  total_rooms  31.968259
0                    longitude  25.623284
4                   population  16.329079
6                median_income   9.043879
1           housing_median_age   8.034782
7       ocean_proximity_INLAND   2.127360
9     ocean_proximity_NEAR BAY   1.343778
10  ocean_proximity_NEAR OCEAN   1.303188
8       ocean_proximity_ISLAND   1.001616
```

In [44]:

```
# 9. Performance metrics ─────────────────────────────────
train_r2 = r2_score(y_train, ols_model.predict(X_train_sm))
test_r2 = r2_score(y_test, y_pred)
rmse = np.sqrt(mean_squared_error(y_test, y_pred))

print("=== Model Performance ===")
print(f"R² (train): {train_r2:.4f}")
print(f"R² (test) : {test_r2:.4f}")
print(f"RMSE (test): {rmse:,.2f}\n")
```

```
=== Model Performance ===
R² (train): 0.6497
R² (test) : 0.6254
RMSE (test): 70,060.52
```

In [45]:

```
#Actual vs Predicted comparison table

compare_df = pd.DataFrame({
```

```
    "Actual": y_test.reset_index(drop=True),
    "Predicted": pd.Series(y_pred)
})
print(compare_df.head(20))
```

```
      Actual      Predicted
0    47700.0   54055.448899
1    45800.0  124225.338937
2   500001.0  255489.379492
3   218600.0  268002.431569
4   278000.0  262769.434816
5   158700.0  139606.303956
6   198200.0  290665.423914
7   157500.0  228264.876375
8   340000.0  256506.785610
9   446600.0  407923.858435
10  123200.0  117648.299242
11  253900.0  177556.028014
12  215100.0   49573.548268
13  220500.0  146896.001320
14  219800.0  249178.850823
15  136200.0   51619.432820
16  178400.0  268970.865935
17  187500.0  206538.013935
18  139800.0  237840.978778
19  137500.0  112245.668928
```

In [46]:
```
#Accuracy %

accuracy_pct = compare_df.corr().iloc[0,1] * 100
print(f"Test-set accuracy ≈ {accuracy_pct:.1f}%")
```

```
Test-set accuracy ≈ 79.1%
```