Importing libraries & Loading the datasets

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

# Load CSVs
admissions = pd.read_csv('admissions.csv')
fatalities = pd.read_csv('fatalities.csv')
metrics = pd.read_csv('metrics.csv')
prescriptions = pd.read_csv('prescriptions.csv')
smokers = pd.read_csv('smokers.csv')
```

data exploration

Display basic info from each dataset

In [2]:
```
admissions.head()
```

Out[2]:

	Year	ICD10 Code	ICD10 Diagnosis	Diagnosis Type	Metric	Sex	Value
0	2014/15	All codes	All admissions	All admissions	Number of admissions	NaN	11011882
1	2014/15	C33-C34 & C00-C14 & C15 & C32 & C53 & C67 & C6...	All diseases which can be caused by smoking	All diseases which can be caused by smoking	Number of admissions	NaN	1713330
2	2014/15	C00-D48	All cancers	All cancers	Number of admissions	NaN	1691035
3	2014/15	J00-J99	All respiratory diseases	All respiratory diseases	Number of admissions	NaN	611002
4	2014/15	I00-I99	All circulatory diseases	All circulatory diseases	Number of admissions	NaN	907157

In [3]:
```
admissions.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2079 entries, 0 to 2078
Data columns (total 7 columns):
 #   Column           Non-Null Count  Dtype 
---  ------           --------------  ----- 
 0   Year             2079 non-null   object
 1   ICD10 Code       2079 non-null   object
 2   ICD10 Diagnosis  2079 non-null   object
 3   Diagnosis Type   2079 non-null   object
 4   Metric           2079 non-null   object
 5   Sex              1386 non-null   object
 6   Value            2078 non-null   object
dtypes: object(7)
memory usage: 113.8+ KB
```

In [4]:
```
fatalities.head()
```

Out[4]:

	Year	ICD10 Code	ICD10 Diagnosis	Diagnosis Type	Metric	Sex	Value
0	2014	All codes	All deaths	All deaths	Number of observed deaths	NaN	459087
1	2014	C33-C34 & C00-C14 & C15 & C32 & C53 & C67 & C6...	All deaths which can be caused by smoking	All deaths which can be caused by smoking	Number of observed deaths	NaN	235820
2	2014	C00-D48	All cancers	All cancers	Number of observed deaths	NaN	136312
3	2014	J00-J99	All respiratory diseases	All respiratory diseases	Number of observed deaths	NaN	61744
4	2014	I00-I99	All circulatory diseases	All circulatory diseases	Number of observed deaths	NaN	126101

In [5]:
```
fatalities.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1749 entries, 0 to 1748
Data columns (total 7 columns):
 #   Column           Non-Null Count  Dtype 
---  ------           --------------  ----- 
 0   Year             1749 non-null   int64 
 1   ICD10 Code       1749 non-null   object
 2   ICD10 Diagnosis  1749 non-null   object
 3   Diagnosis Type   1749 non-null   object
 4   Metric           1749 non-null   object
 5   Sex              1166 non-null   object
 6   Value            1749 non-null   object
dtypes: int64(1), object(6)
memory usage: 95.8+ KB
```

In [6]:
```
metrics.head()
```

Out[6]:

	Year	Tobacco Price\nIndex	Retail Prices\nIndex	Tobacco Price Index Relative to Retail Price Index	Real Households' Disposable Income	Affordability of Tobacco Index	Household Expenditure on Tobacco
0	2015	1294.3	386.7	334.7	196.4	58.7	19252.0
1	2014	1226.0	383.0	320.1	190.0	59.4	19411.0
2	2013	1139.3	374.2	304.5	190.3	62.5	18683.0
3	2012	1057.8	363.1	291.3	192.9	66.2	18702.0
4	2011	974.9	351.9	277.1	189.3	68.3	18217.0

In [7]:
```
metrics.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 36 entries, 0 to 35
Data columns (total 9 columns):
 #   Column                                               Non-Null Count  Dtype
---  ------                                               --------------  -----
 0   Year                                                 36 non-null     int64
 1   Tobacco Price
Index                                      36 non-null     float64
 2   Retail Prices
Index                                      36 non-null     float64
 3   Tobacco Price Index Relative to Retail Price Index   36 non-null     float
64
 4   Real Households' Disposable Income                   36 non-null     float
64
 5   Affordability of Tobacco Index                       36 non-null     float
64
 6   Household Expenditure on Tobacco                     31 non-null     float
64
 7   Household Expenditure Total                          31 non-null     float
64
 8   Expenditure on Tobacco as a Percentage of Expenditure  31 non-null     float
64
dtypes: float64(8), int64(1)
memory usage: 2.7 KB
```

In [8]:
```
prescriptions.head()
```

Out[8]:

	Year	All Pharmacotherapy Prescriptions	Nicotine Replacement Therapy (NRT) Prescriptions	Bupropion (Zyban) Prescriptions	Varenicline (Champix) Prescriptions	Net Ingrec Cost o Pharmacothera
0	2014/15	1348	766	21	561.0	3
1	2013/14	1778	1059	22	697.0	4
2	2012/13	2203	1318	26	859.0	5
3	2011/12	2532	1545	30	957.0	6
4	2010/11	2564	1541	36	987.0	6

In [9]:
```
prescriptions.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 11 entries, 0 to 10
Data columns (total 9 columns):
 #   Column                                                Non-Null Count
Dtype
---  ------                                                --------------
-----
 0   Year                                                  11 non-null
object
 1   All Pharmacotherapy Prescriptions                     11 non-null
int64
 2   Nicotine Replacement Therapy (NRT) Prescriptions      11 non-null
int64
 3   Bupropion (Zyban) Prescriptions                       11 non-null
int64
 4   Varenicline (Champix) Prescriptions                   9 non-null
float64
 5   Net Ingredient Cost of All Pharmacotherapies          11 non-null
int64
 6   Net Ingredient Cost of Nicotine Replacement Therapies (NRT)  11 non-null
int64
 7   Net Ingredient Cost of Bupropion (Zyban)              11 non-null
int64
 8   Net Ingredient Cost of Varenicline (Champix)          9 non-null
float64
dtypes: float64(2), int64(6), object(1)
memory usage: 924.0+ bytes
```

In [10]:
```
smokers.head()
```

Out[10]:

	Year	Method	Sex	16 and Over	16-24	25-34	35-49	50-59	60 and Over
0	1974	Unweighted	NaN	46	44	51	52	50	33
1	1976	Unweighted	NaN	42	42	45	48	48	30
2	1978	Unweighted	NaN	40	39	45	45	45	30
3	1980	Unweighted	NaN	39	37	46	44	45	29
4	1982	Unweighted	NaN	35	35	38	39	41	27

In [11]:
```
smokers.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 84 entries, 0 to 83
Data columns (total 9 columns):
 #   Column       Non-Null Count  Dtype 
---  ------       --------------  ----- 
 0   Year         84 non-null     int64 
 1   Method       84 non-null     object
 2   Sex          56 non-null     object
 3   16 and Over  84 non-null     int64 
 4   16-24        84 non-null     int64 
 5   25-34        84 non-null     int64 
 6   35-49        84 non-null     int64 
 7   50-59        84 non-null     int64 
 8   60 and Over  84 non-null     int64 
dtypes: int64(7), object(2)
memory usage: 6.0+ KB
```

data cleaning

In [12]:
```
# Standardize 'Year' formats
def standardize_year(col):
    # Handles "2014/15" -> 2014, "2014" -> 2014
    # If the year is in "YYYY/YY" format, take the first 4 digits
    return col.astype(str).str[:4].astype(int)

admissions["Year"] = standardize_year(admissions["Year"])
fatalities["Year"] = standardize_year(fatalities["Year"])
prescriptions["Year"] = standardize_year(prescriptions["Year"])
smokers["Year"] = standardize_year(smokers["Year"])
# Metrics years are already int
```

Handling missing values

In [13]:
```
# Count
admissions.isnull().sum()
fatalities.isnull().sum()
metrics.isnull().sum()
prescriptions.isnull().sum()
smokers.isnull().sum()
```

Out[13]:
```
Year             0
Method           0
Sex             28
16 and Over      0
16-24            0
25-34            0
35-49            0
50-59            0
60 and Over      0
dtype: int64
```

In [14]:
```
admissions.replace([".", ""], np.nan, inplace=True)
fatalities.replace([".", ""], np.nan, inplace=True)
prescriptions.replace([".", ""], np.nan, inplace=True)
smokers.replace([".", ""], np.nan, inplace=True)
metrics.replace([".", ""], np.nan, inplace=True)

for df in [admissions, fatalities, prescriptions, smokers, metrics]:
    for col in df.columns:
        df[col] = pd.to_numeric(df[col], errors="ignore")
```

```
C:\Users\asus\AppData\Local\Temp\ipykernel_15692\2333883816.py:9: FutureWarning:
errors='ignore' is deprecated and will raise in a future version. Use to_numeric
without passing `errors` and catch exceptions explicitly instead
  df[col] = pd.to_numeric(df[col], errors="ignore")
```

In [15]:
```
# Check for duplicates
print("Admissions duplicates:", admissions.duplicated().sum())
print("Fatalities duplicates:", fatalities.duplicated().sum())
print("Metrics duplicates:", metrics.duplicated().sum())
print("Prescriptions duplicates:", prescriptions.duplicated().sum())
print("Smokers duplicates:", smokers.duplicated().sum())
```

```
Admissions duplicates: 0
Fatalities duplicates: 0
Metrics duplicates: 0
Prescriptions duplicates: 0
Smokers duplicates: 0
```

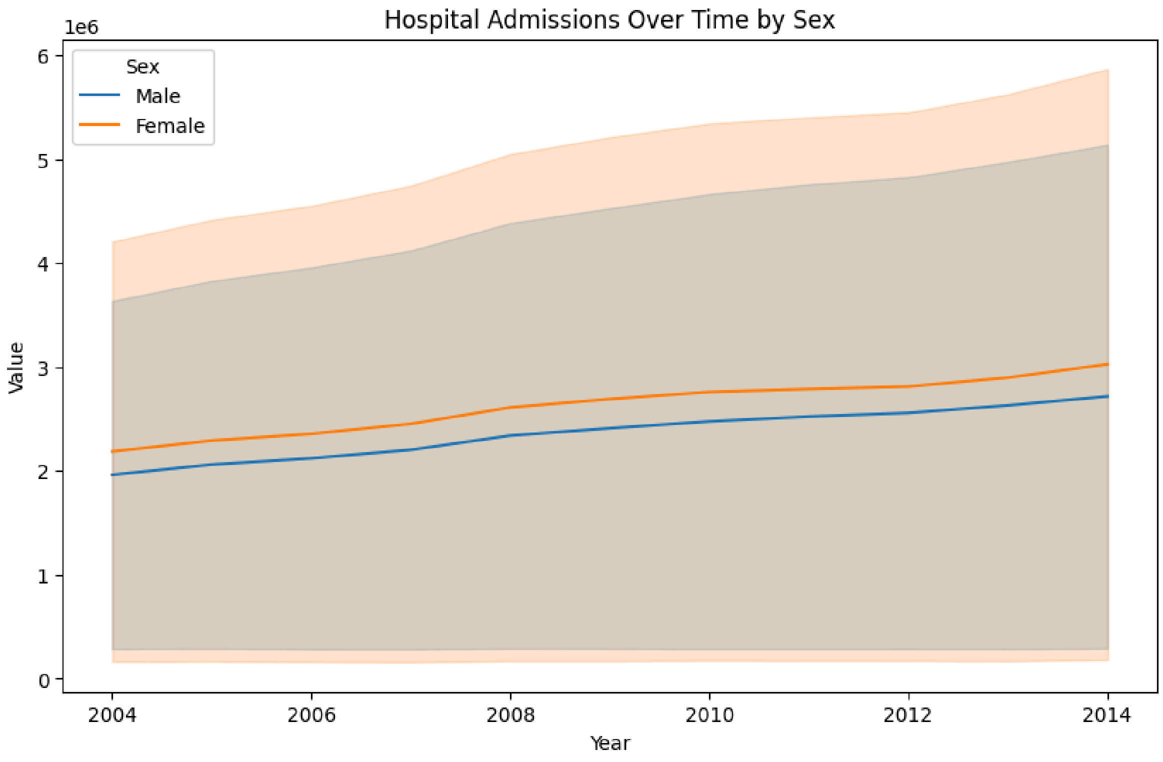
In [16]:
```
# Output cleaned dataframes
admissions_clean = admissions.copy()
fatalities_clean = fatalities.copy()
metrics_clean = metrics.copy()
prescriptions_clean = prescriptions.copy()
smokers_clean = smokers.copy()
```

In [17]:
```
# Save cleaned datasets
admissions_clean.to_csv("admissions_clean.csv", index=False)
fatalities_clean.to_csv("fatalities_clean.csv", index=False)
metrics_clean.to_csv("metrics_clean.csv", index=False)
prescriptions_clean.to_csv("prescriptions_clean.csv", index=False)
smokers_clean.to_csv("smokers_clean.csv", index=False)

print("Data loading and initial cleaning complete.")
```

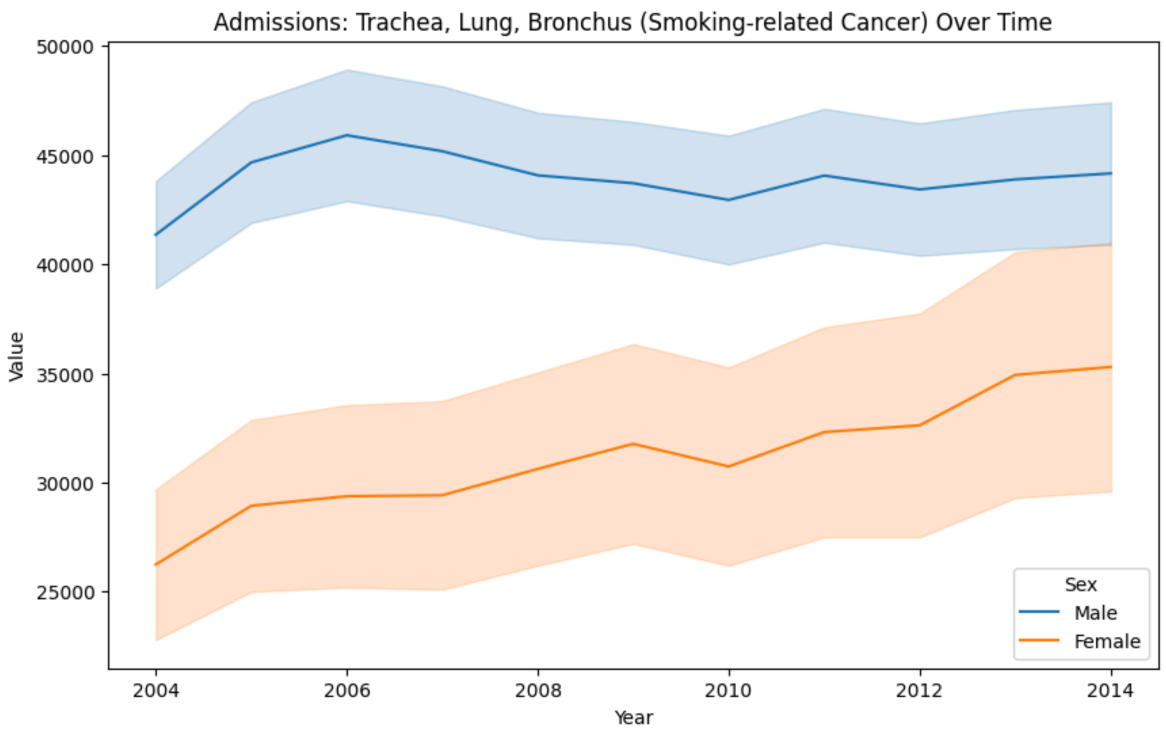
```
Data loading and initial cleaning complete.
```

EDA on the datasets

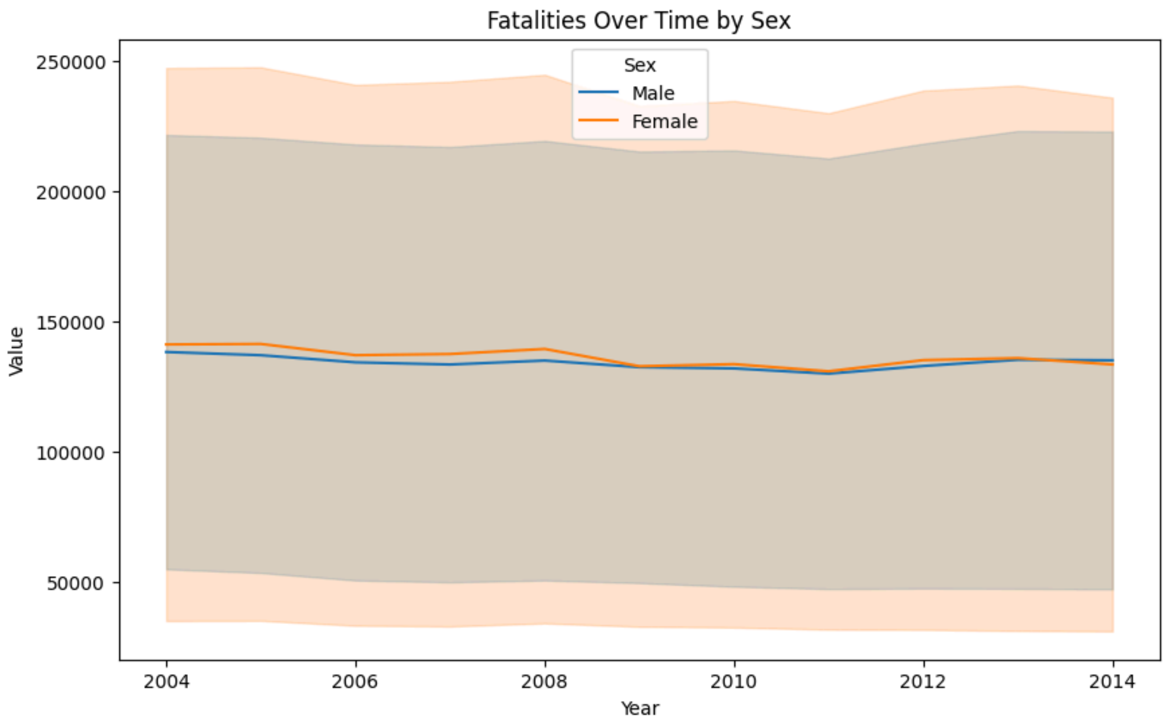
In [18]:
```
## Admissions
plt.figure(figsize=(10,6))
sns.lineplot(data=admissions_clean[admissions_clean['Diagnosis Type'] == 'All ad
             x='Year', y='Value', hue='Sex')
plt.title("Hospital Admissions Over Time by Sex")
plt.show()
```



In [19]:
```
## Admissions: Time trend for selected diagnosis
plt.figure(figsize=(10,6))
sns.lineplot(
    data=admissions_clean[(admissions_clean['ICD10 Diagnosis'] == "Trachea, Lung
    x='Year', y='Value', hue='Sex'
)
plt.title("Admissions: Trachea, Lung, Bronchus (Smoking-related Cancer) Over Tim
plt.show()
```

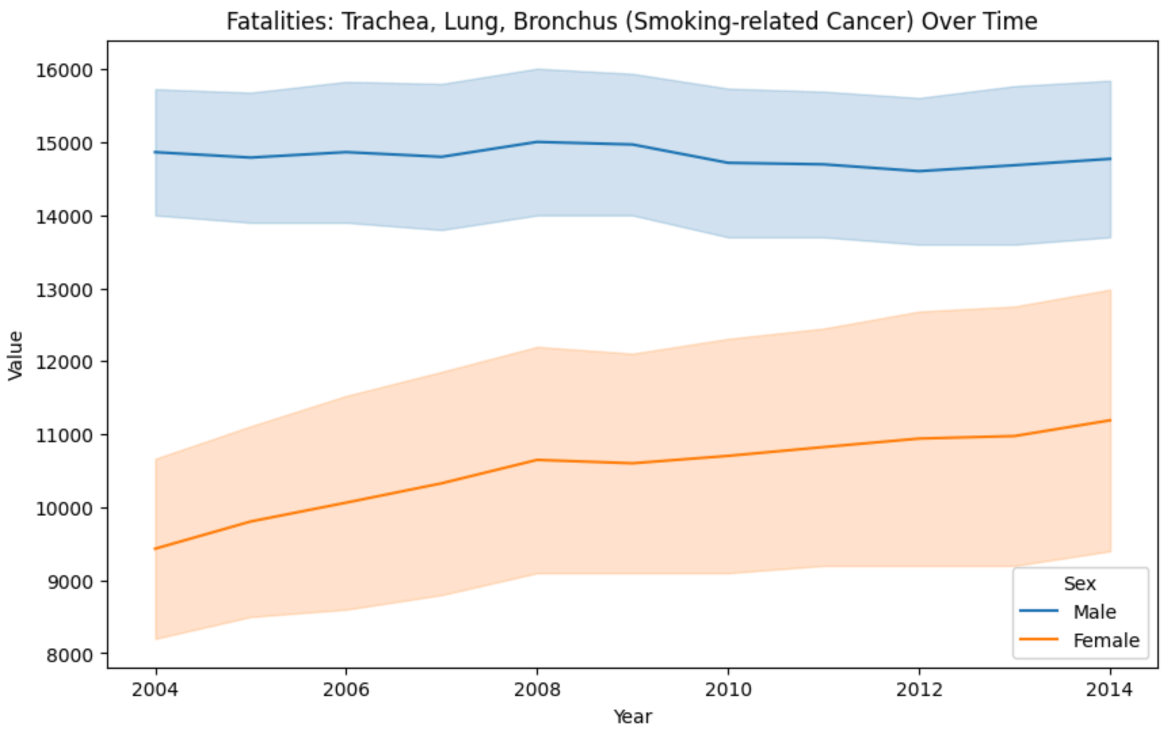


In [20]:
```
## Fatalities
plt.figure(figsize=(10,6))
sns.lineplot(data=fatalities_clean[fatalities_clean['Diagnosis Type'] == 'All de
             x='Year', y='Value', hue='Sex')
plt.title("Fatalities Over Time by Sex")
plt.show()
```

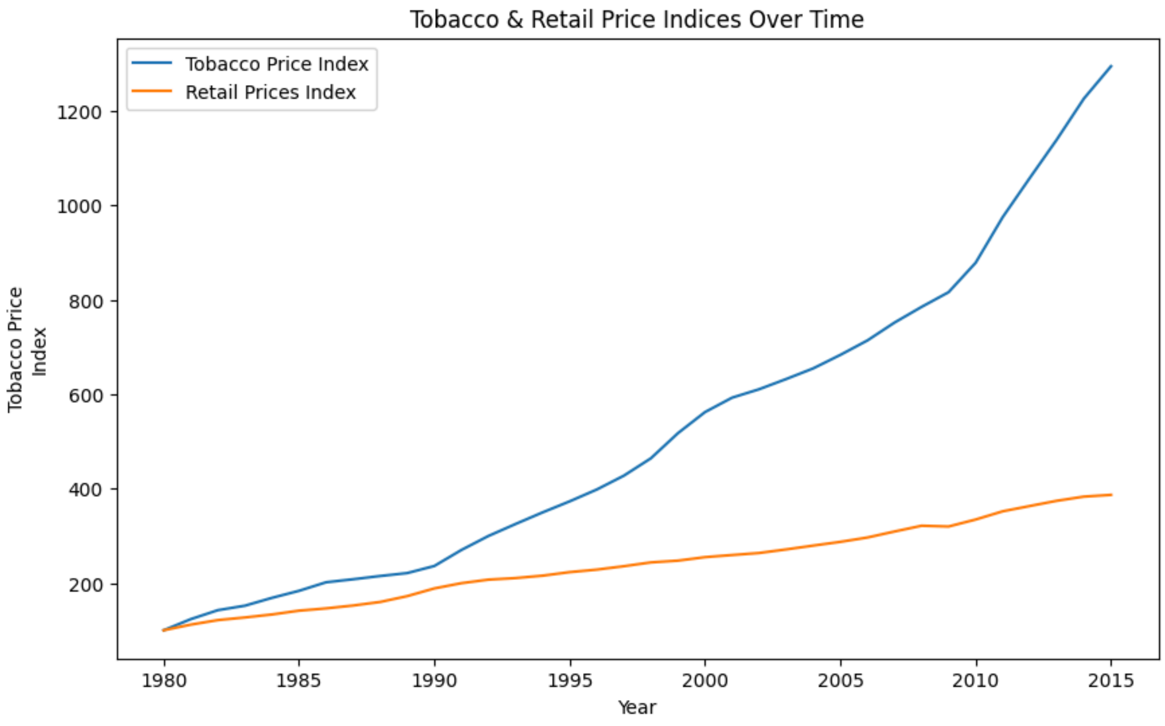


In [21]:
```
## Fatalities: Time trend for selected diagnosis
plt.figure(figsize=(10,6))
sns.lineplot(
    data=fatalities_clean[(fatalities_clean['ICD10 Diagnosis'] == "Trachea, Lung
    x='Year', y='Value', hue='Sex'
)
```

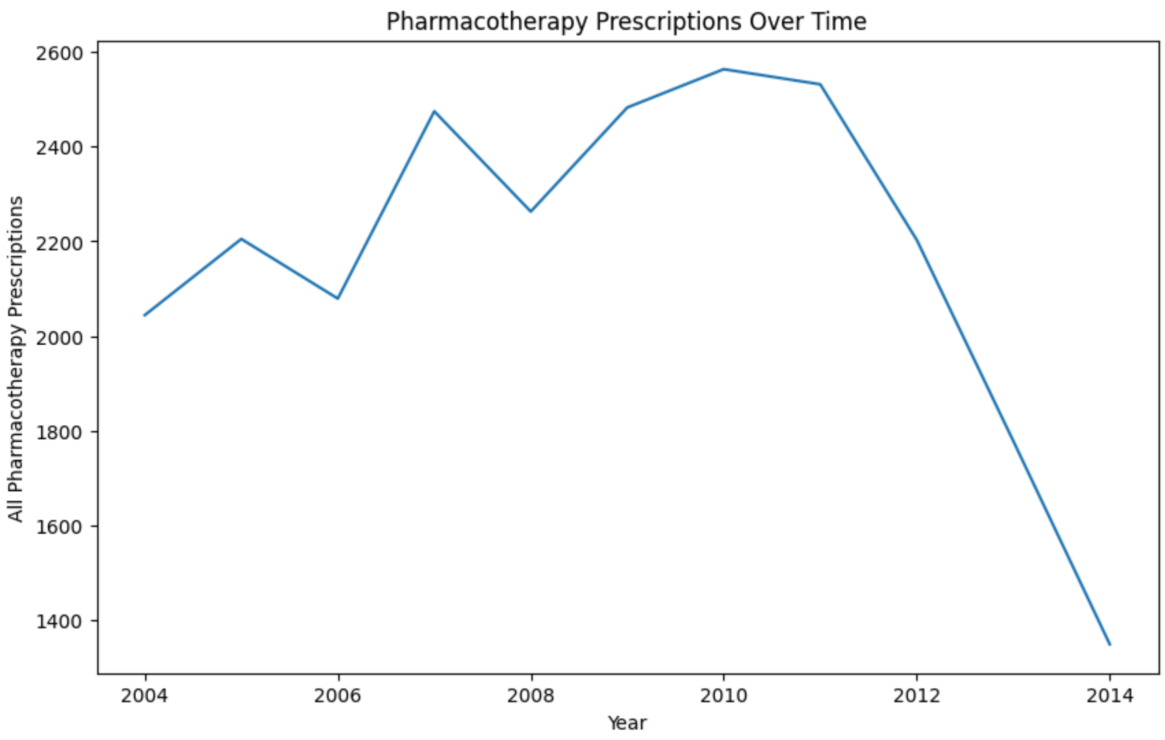
```
plt.title("Fatalities: Trachea, Lung, Bronchus (Smoking-related Cancer) Over Tim
plt.show()
```



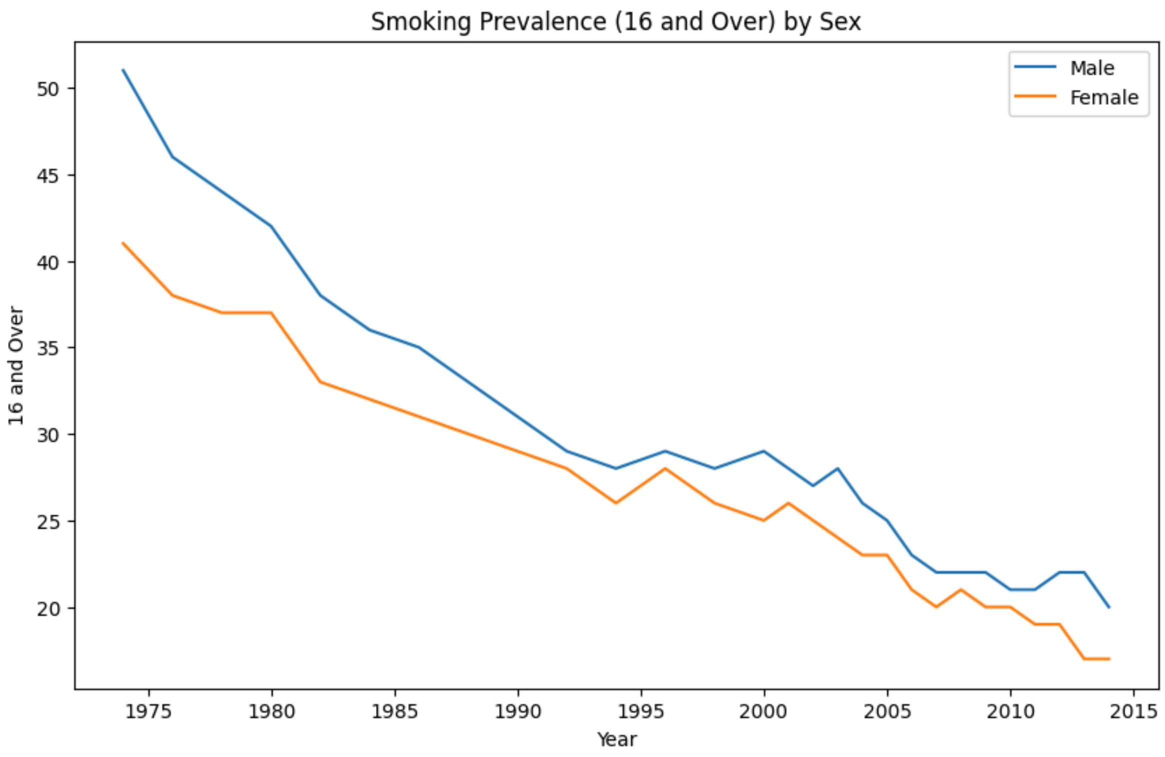
In [22]:
```
## Metrics
plt.figure(figsize=(10,6))
sns.lineplot(data=metrics_clean, x='Year', y='Tobacco Price\nIndex', label='Toba
sns.lineplot(data=metrics_clean, x='Year', y='Retail Prices\nIndex', label='Reta
plt.legend()
plt.title("Tobacco & Retail Price Indices Over Time")
plt.show()
```



In [23]:
```
## Prescriptions
plt.figure(figsize=(10,6))
sns.lineplot(data=prescriptions_clean, x='Year', y='All Pharmacotherapy Prescrip
plt.title("Pharmacotherapy Prescriptions Over Time")
plt.show()
```



In [24]:
```
## Smokers
plt.figure(figsize=(10,6))
sns.lineplot(data=smokers_clean[smokers_clean['Sex']=="Male"], x='Year', y='16 a
sns.lineplot(data=smokers_clean[smokers_clean['Sex']=="Female"], x='Year', y='16
plt.title("Smoking Prevalence (16 and Over) by Sex")
plt.legend()
plt.show()
```



merging the datasets

In [25]:
```
# loading cleaned data
admissions = pd.read_csv("admissions_clean.csv")
fatalities = pd.read_csv("fatalities_clean.csv")
metrics = pd.read_csv("metrics_clean.csv")
prescriptions = pd.read_csv("prescriptions_clean.csv")
smokers = pd.read_csv("smokers_clean.csv")
```

In [26]:
```
# 1. Aggregate admissions and fatalities to year/sex/diagnosis level
admissions_agg = admissions.groupby(['Year', 'Sex', 'ICD10 Diagnosis', 'Diagnosi
fatalities_agg = fatalities.groupby(['Year', 'Sex', 'ICD10 Diagnosis', 'Diagnosi

# 2. Aggregate smokers to year/sex level (e.g., overall prevalence for '16 and O
smokers_agg = smokers.groupby(['Year', 'Sex'])['16 and Over'].mean().reset_index
smokers_agg.rename(columns={'16 and Over': 'Smoking Prevalence'}, inplace=True)

# 3. Metrics and prescriptions: only by year, so merge on year.
# If you want sex-specific analysis, merge those values to all sex categories pe
metrics_agg = metrics.copy()
prescriptions_agg = prescriptions.copy()
```

In [27]:
```
# 4. Merge all together (outer join to preserve all possible combinations)
df = admissions_agg.merge(fatalities_agg, on=['Year', 'Sex', 'ICD10 Diagnosis',
df = df.merge(smokers_agg, on=['Year', 'Sex'], how='left')
df = df.merge(metrics_agg, on='Year', how='left')
df = df.merge(prescriptions_agg, on='Year', how='left')
df.head()
```

Out[27]:

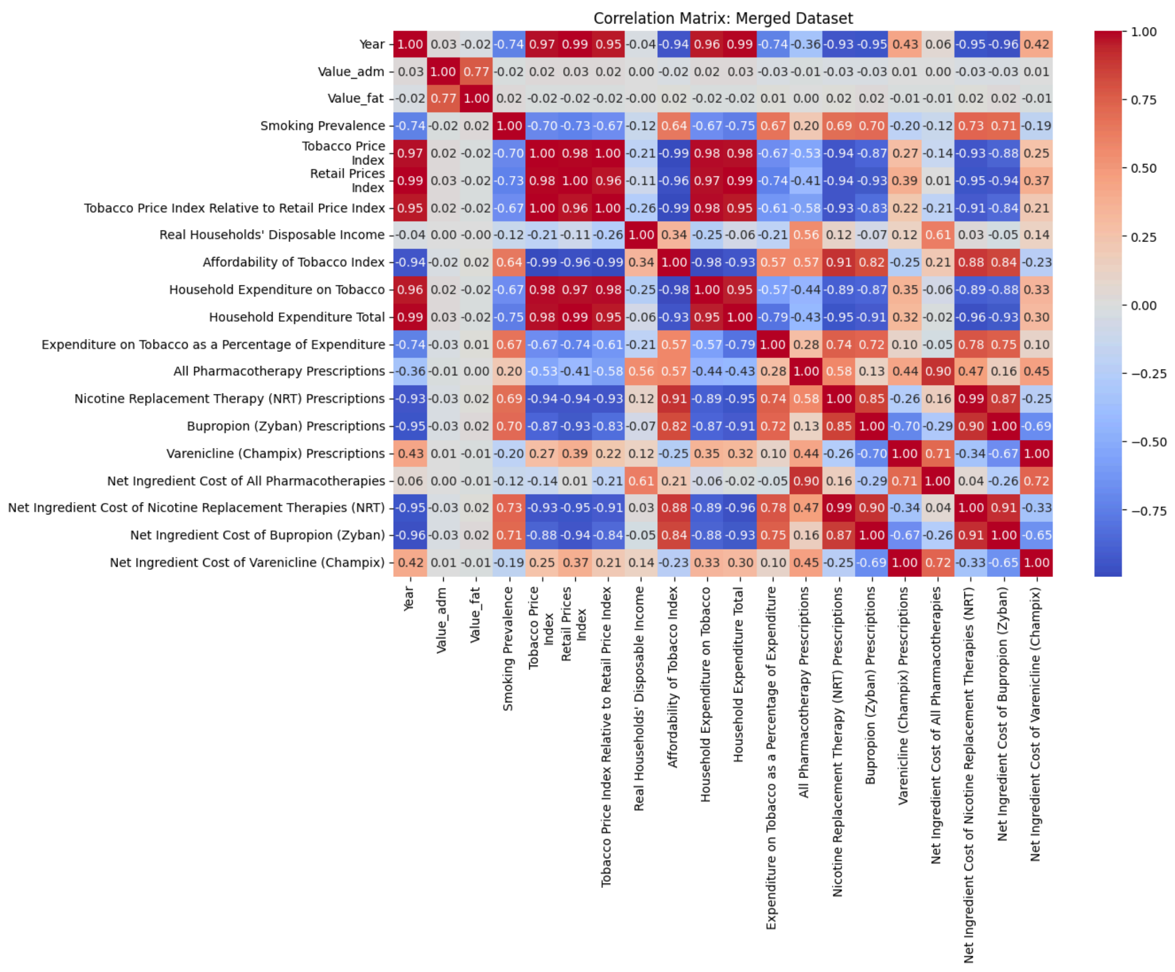
	Year	Sex	ICD10 Diagnosis	Diagnosis Type	Value_adm	Value_fat	Smoking Prevalence	Tobacc Price\nIndex
0	2004	Female	Age Related Cataract 45+	Other diseases which can be caused by smoking	87824.0	NaN	23.0	654.6
1	2004	Female	All admissions	All admissions	4373273.0	NaN	23.0	654.6
2	2004	Female	All cancers	All cancers	646853.0	74122.0	23.0	654.6
3	2004	Female	All circulatory diseases	All circulatory diseases	403117.0	102726.0	23.0	654.6
4	2004	Female	All deaths	All deaths	NaN	282255.0	23.0	654.6

5 rows × 23 columns

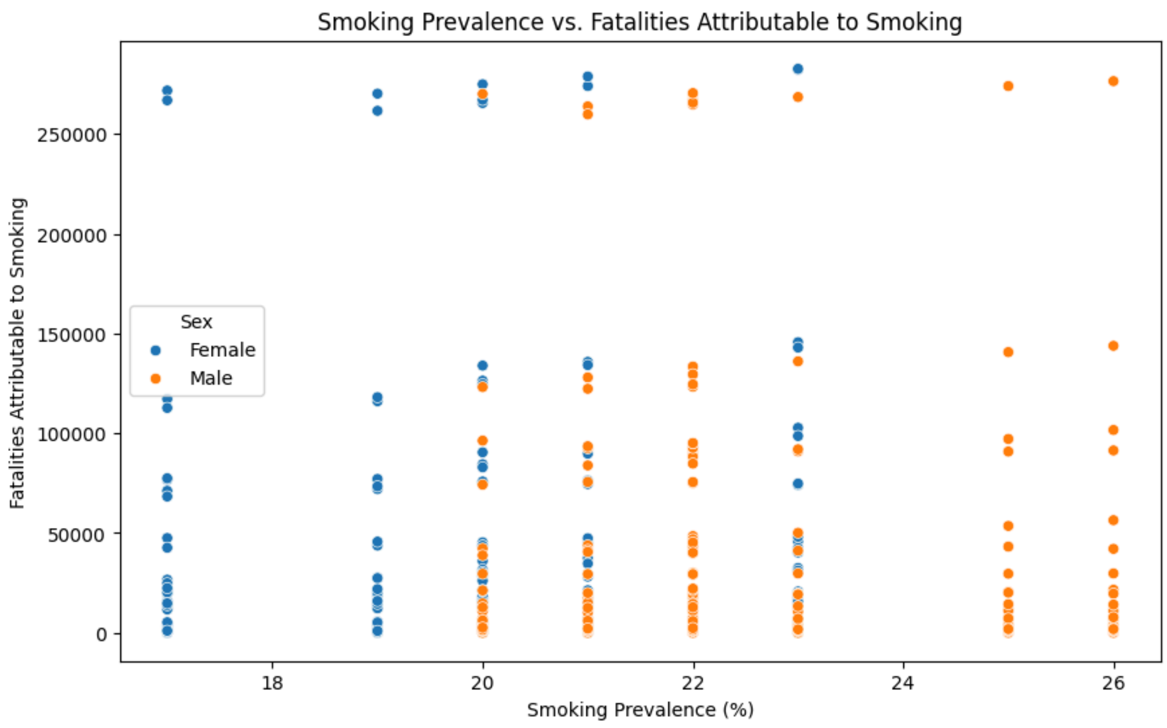
In [28]:
```
# Save merged dataset for modeling
df.to_csv("merged_dataset.csv", index=False)
```

EDA on the Merged dataset

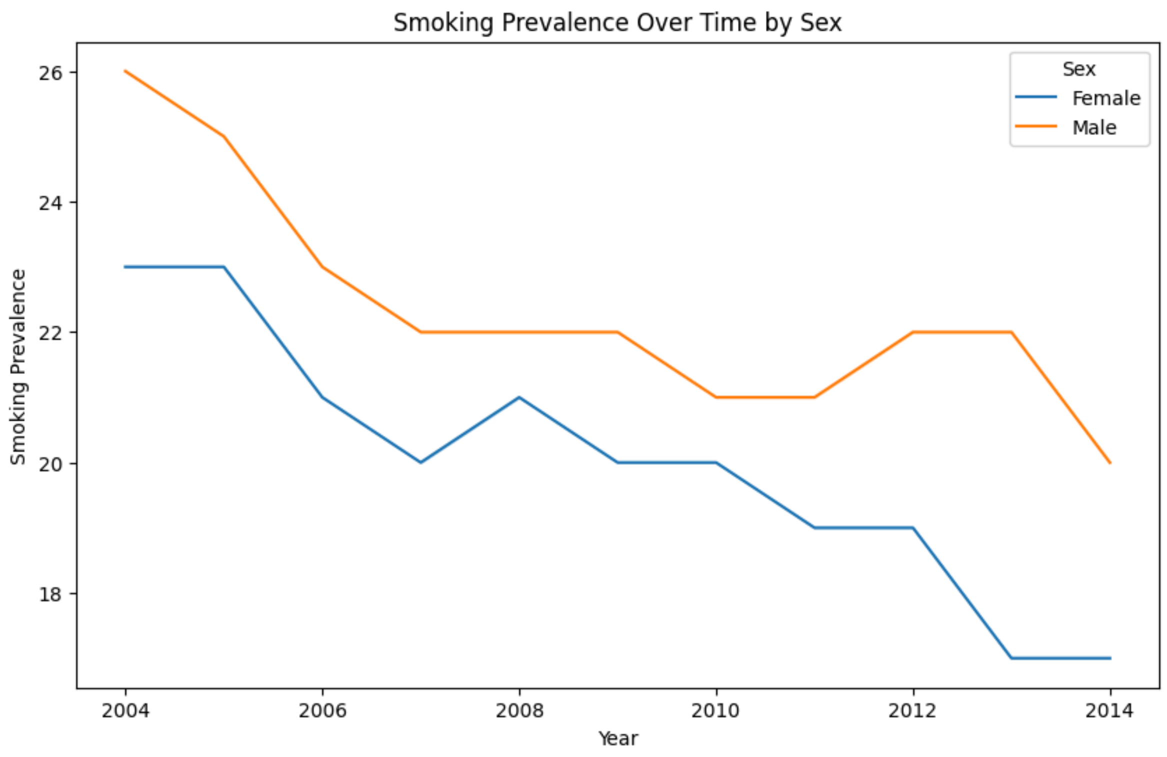
In [29]:
```
merged = pd.read_csv('merged_dataset.csv')
## Correlation heatmap
corr = merged.select_dtypes(include='number').corr()
plt.figure(figsize=(12,8))
sns.heatmap(corr, annot=True, fmt='.2f', cmap='coolwarm')
plt.title("Correlation Matrix: Merged Dataset")
plt.show()
```



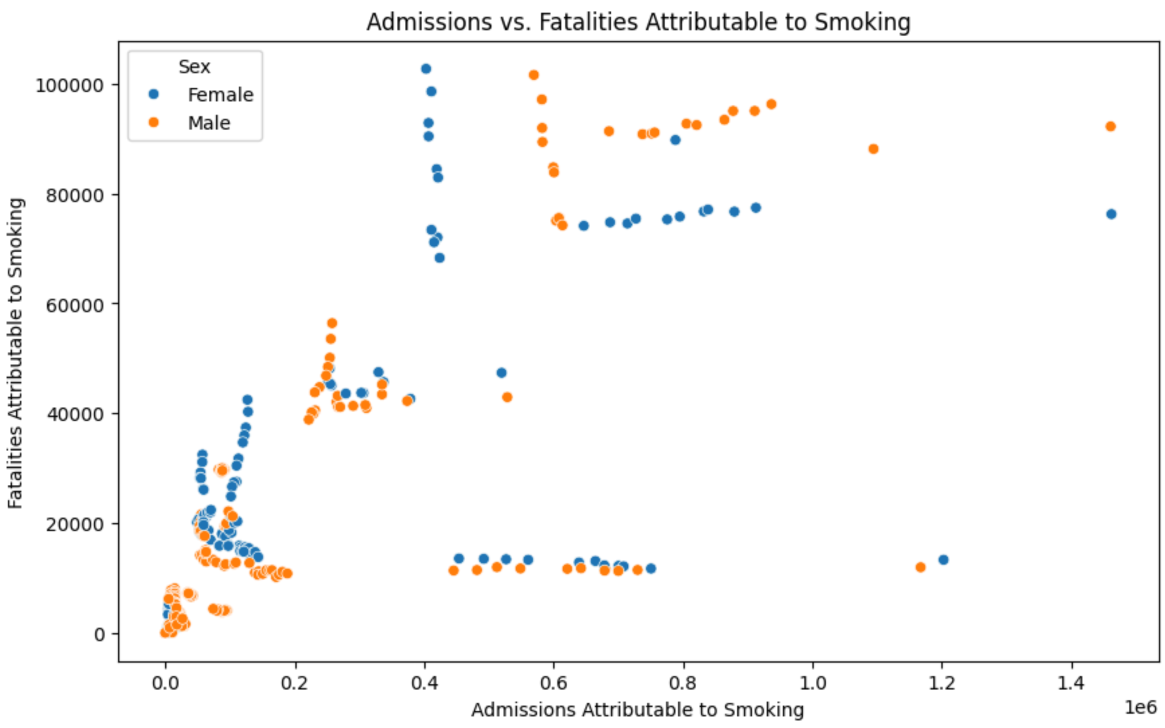
```
In [30]: ## Smoking prevalence vs. deaths attributable to smoking
         plt.figure(figsize=(10,6))
         sns.scatterplot(data=merged, x='Smoking Prevalence', y='Value_fat',
                         hue='Sex')
         plt.title("Smoking Prevalence vs. Fatalities Attributable to Smoking")
         plt.xlabel("Smoking Prevalence (%)")
         plt.ylabel("Fatalities Attributable to Smoking")
         plt.show()
```



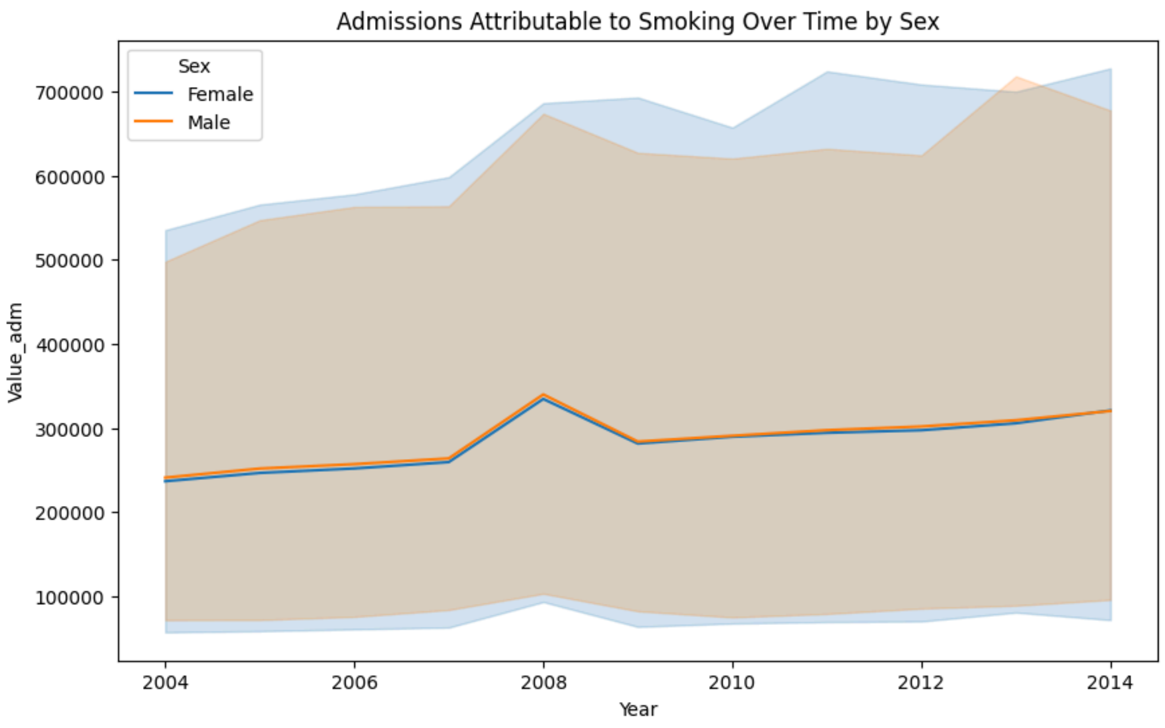
In [31]:
```
## Time trend: Smoking prevalence over time
plt.figure(figsize=(10,6))
sns.lineplot(data=merged, x='Year', y='Smoking Prevalence', hue='Sex')
plt.title("Smoking Prevalence Over Time by Sex")
plt.show()
```



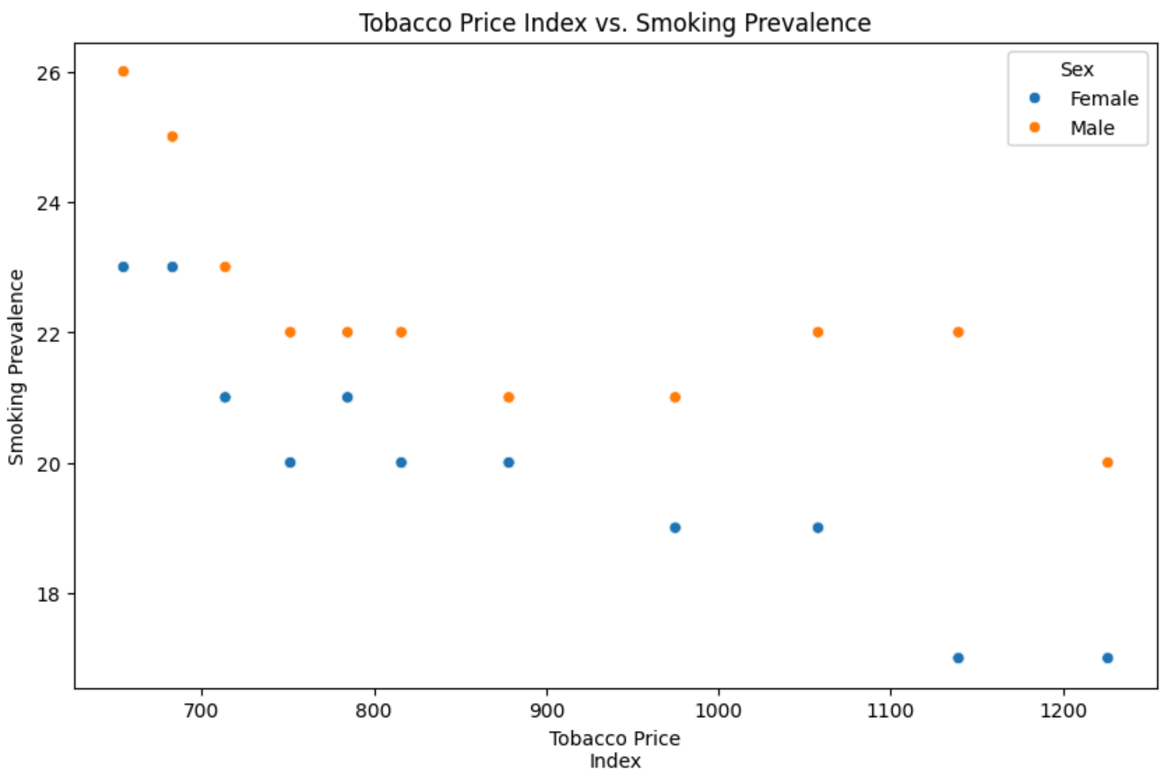
In [32]:
```
## Admissions vs. deaths attributable to smoking
plt.figure(figsize=(10,6))
sns.scatterplot(data=merged, x='Value_adm', y='Value_fat', hue='Sex')
plt.title("Admissions vs. Fatalities Attributable to Smoking")
plt.xlabel("Admissions Attributable to Smoking")
plt.ylabel("Fatalities Attributable to Smoking")
plt.show()
```



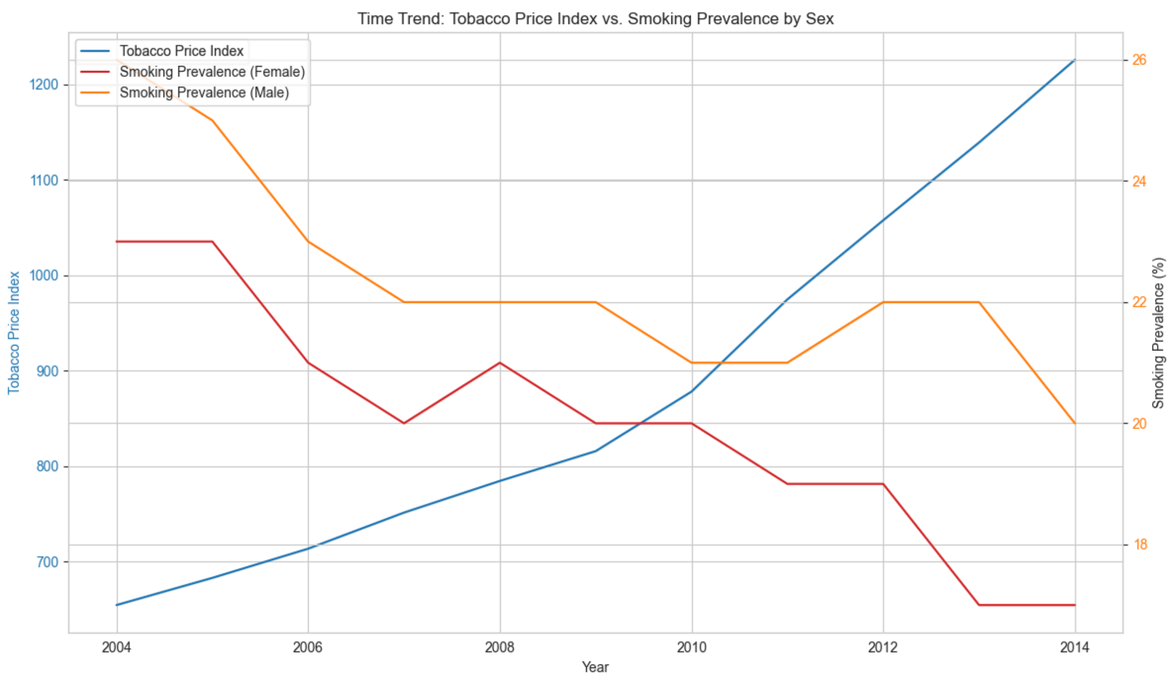
In [33]:
```
## Time trend: Admissions attributable to smoking over time
plt.figure(figsize=(10,6))
sns.lineplot(data=merged, x='Year', y='Value_adm', hue='Sex')
plt.title("Admissions Attributable to Smoking Over Time by Sex")
plt.show()
```



In [34]:
```
## Tobacco price vs. smoking prevalence
plt.figure(figsize=(10,6))
sns.scatterplot(data=merged, x='Tobacco Price\nIndex', y='Smoking Prevalence', h
plt.title("Tobacco Price Index vs. Smoking Prevalence")
plt.show()
```



In [35]:
```python
# Plot time trends for both variables on dual axes
sexes = merged['Sex'].dropna().unique()
# Assign colors explicitly to each sex for clarity
sex_color_map = {'Male': 'tab:orange', 'Female': 'tab:red'}
# Fallback if there are unexpected categories
for s in sexes:
    if s not in sex_color_map:
        sex_color_map[s] = 'tab:green'

plt.figure(figsize=(12,7))
sns.set_style("whitegrid")

# Plot Tobacco Price Index (left y-axis)
ax1 = plt.gca()
sns.lineplot(
    data=merged, x='Year', y='Tobacco Price\nIndex',
    color='tab:blue', label='Tobacco Price Index', ax=ax1
)
ax1.set_ylabel('Tobacco Price Index', color='tab:blue')
ax1.tick_params(axis='y', labelcolor='tab:blue')

# Plot Smoking Prevalence by Sex (right y-axis)
ax2 = ax1.twinx()
for sex in sexes:
    subset = merged[merged['Sex'] == sex]
    sns.lineplot(
        data=subset, x='Year', y='Smoking Prevalence',
        ax=ax2, color=sex_color_map[sex], label=f"Smoking Prevalence ({sex})"
    )
ax2.set_ylabel('Smoking Prevalence (%)')
ax2.tick_params(axis='y', labelcolor='tab:orange')

# Combine legends from both axes
lines_labels = [ax.get_legend_handles_labels() for ax in [ax1, ax2]]
lines, labels = [sum(lol, []) for lol in zip(*lines_labels)]
plt.legend(lines, labels, loc='upper left')
plt.title('Time Trend: Tobacco Price Index vs. Smoking Prevalence by Sex')
plt.tight_layout()
plt.show()
```



Feature engineering

```
In [36]: merged = pd.read_csv("merged_dataset.csv")

         # 1. Year-over-year change in Smoking Prevalence
         merged['Smoking_Prevalence_YoY'] = merged.groupby('Sex')['Smoking Prevalence'].d

         # 2. Death Rate (per admission)
         merged['Death_Rate'] = merged['Value_fat'] / (merged['Value_adm'] + 1e-6)

         # 3. Interaction Term: Smoking Prevalence * Tobacco Price Index
         merged['SmokingPrice_Interaction'] = merged['Smoking Prevalence'] * merged['Toba

         # 4. Categorize years by policy era
         merged['Policy_Era'] = np.where(merged['Year'] < 2010, 'Pre-2010', 'Post-2010')

         # 5. Fill missing values (simple imputation)
         merged.fillna(method='ffill', inplace=True)

         # 6. One-hot encode categorical variables
         merged = pd.get_dummies(merged, columns=['Sex', 'Policy_Era'], drop_first=True)

         # Save engineered dataset
         merged.to_csv("merged_featured.csv", index=False)

         print("Feature engineering complete. Sample of new features:")
         merged.head()
```

```
Feature engineering complete. Sample of new features:
```

```
C:\Users\asus\AppData\Local\Temp\ipykernel_15692\3620466036.py:16: FutureWarning: 
DataFrame.fillna with 'method' is deprecated and will raise in a future version. 
Use obj.ffill() or obj.bfill() instead.
  merged.fillna(method='ffill', inplace=True)
```

Out[36]:

	Year	ICD10 Diagnosis	Diagnosis Type	Value_adm	Value_fat	Smoking Prevalence	Tobacco Price\nIndex	Prices\
0	2004	Age Related Cataract 45+	Other diseases which can be caused by smoking	87824.0	NaN	23.0	654.6	
1	2004	All admissions	All admissions	4373273.0	NaN	23.0	654.6	
2	2004	All cancers	All cancers	646853.0	74122.0	23.0	654.6	
3	2004	All circulatory diseases	All circulatory diseases	403117.0	102726.0	23.0	654.6	
4	2004	All deaths	All deaths	403117.0	282255.0	23.0	654.6	

5 rows × 27 columns

Model Training and Evaluation for Tobacco Use and Mortality Data

In [37]:
```python
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, r2_score
```

In [51]:
```python
# Load engineered dataset
data = pd.read_csv("merged_featured.csv")

df = df.dropna(subset=["Value_fat"])

# Separate features and target
X = df.drop(["Year", "Value_fat"], axis=1)
y = df["Value_fat"]
```

In [52]:
```python
# One-hot encode categorical columns
X = pd.get_dummies(X, drop_first=True)
```

```
# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_
```

In [53]:
```
# Impute numeric NaNs
imputer = SimpleImputer(strategy="mean")
X_train_imputed = imputer.fit_transform(X_train)
X_test_imputed = imputer.transform(X_test)
```

In [54]:
```
# Fit Linear Regression
lr = LinearRegression()
lr.fit(X_train_imputed, y_train)
y_pred_lr = lr.predict(X_test_imputed)
```

In [55]:
```
# Fit Random Forest
rf = RandomForestRegressor(n_estimators=100, random_state=42)
rf.fit(X_train_imputed, y_train)
y_pred_rf = rf.predict(X_test_imputed)
```

In [58]:
```
# Evaluation
from sklearn.metrics import mean_squared_error, r2_score
import numpy as np
lr_rmse = np.sqrt(mean_squared_error(y_test, y_pred_lr))
rf_rmse = np.sqrt(mean_squared_error(y_test, y_pred_rf))

lr_r2 = r2_score(y_test, y_pred_lr)
rf_r2 = r2_score(y_test, y_pred_rf)

print("Linear Regression RMSE:", lr_rmse)
print("Random Forest RMSE:", rf_rmse)
print("Linear Regression R²:", lr_r2)
print("Random Forest R²:", rf_r2)
```

```
Linear Regression RMSE: 3967.648195334058
Random Forest RMSE: 2466.6074507367907
Linear Regression R²: 0.9971116808418315
Random Forest R²: 0.9988837050889491
```

Random forest classifier

In [68]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import accuracy_score, confusion_matrix, roc_auc_score, cla
```

In [69]:
```
data = pd.read_csv("merged_featured.csv")
target = 'Value_fat'
if pd.api.types.is_numeric_dtype(data[target]):
    median = data[target].median()
    data['target_bin'] = (data[target] > median).astype(int)
    target = 'target_bin'

features = [col for col in data.columns if col not in ['Year', 'Value_fat', 'tar

X = data[features]
y = data[target]

X_encoded = pd.get_dummies(X)
```

```
# Train/Test split
X_train, X_test, y_train, y_test = train_test_split(X_encoded, y, test_size=0.2,
```

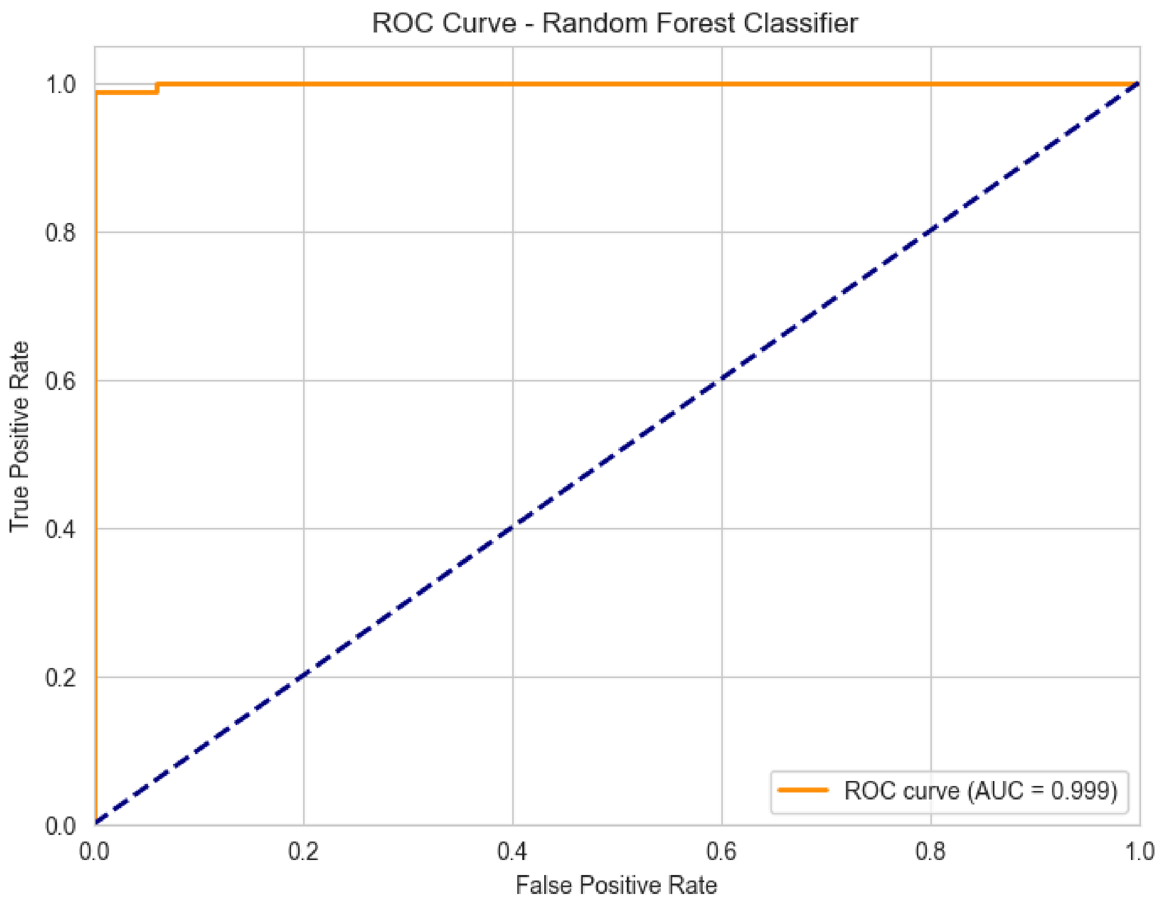
In [70]:
```
# Impute missing
imputer = SimpleImputer(strategy='mean')
X_train_imputed = imputer.fit_transform(X_train)
X_test_imputed = imputer.transform(X_test)

# Train Random Forest Classifier
rf_clf = RandomForestClassifier(n_estimators=100, random_state=42)
rf_clf.fit(X_train_imputed, y_train)

#  Predict labels + probabilities
y_pred_rf = rf_clf.predict(X_test_imputed)
y_prob_rf = rf_clf.predict_proba(X_test_imputed)
```

In [71]:
```
# Accuracy
acc_rf = accuracy_score(y_test, y_pred_rf)
print("Random Forest Accuracy:", acc_rf)

# Confusion Matrix
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred_rf))

# Classification Report
print("Classification Report:\n", classification_report(y_test, y_pred_rf))

# ROC AUC
if len(rf_clf.classes_) > 2:
    roc_auc_rf = roc_auc_score(y_test, y_prob_rf, multi_class='ovr', labels=rf_c
else:
    roc_auc_rf = roc_auc_score(y_test, y_prob_rf[:, 1])
print(" ROC AUC:", roc_auc_rf)
```

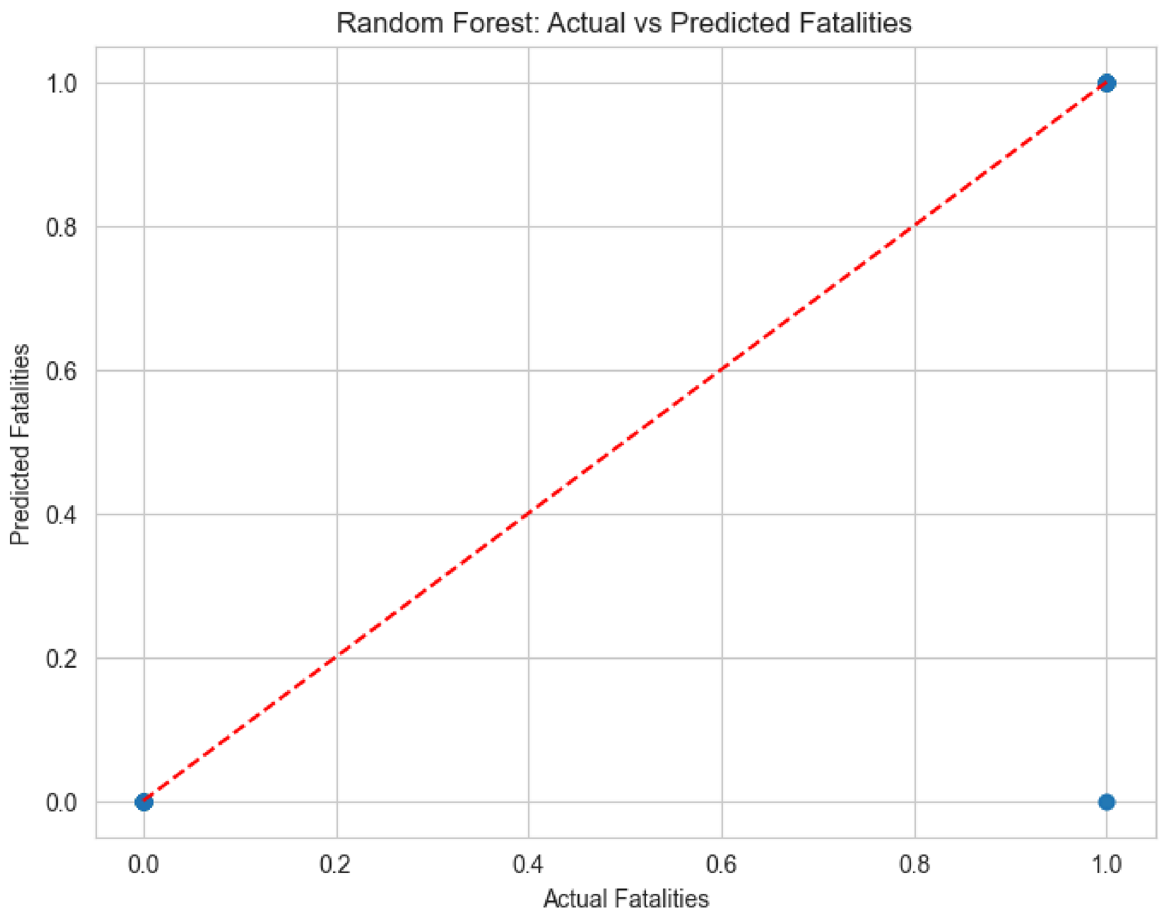
```
Random Forest Accuracy: 0.98
Confusion Matrix:
 [[67  0]
 [ 3 80]]
Classification Report:
               precision    recall  f1-score   support

           0       0.96      1.00      0.98        67
           1       1.00      0.96      0.98        83

    accuracy                           0.98       150
   macro avg       0.98      0.98      0.98       150
weighted avg       0.98      0.98      0.98       150

 ROC AUC: 0.9992807049091891
```

In [75]:
```
from sklearn.metrics import roc_curve, auc

y_prob_rf = rf_clf.predict_proba(X_test_imputed)[:, 1]

# Compute FPR, TPR
fpr, tpr, thresholds = roc_curve(y_test, y_prob_rf)
roc_auc = auc(fpr, tpr)
# Plot ROC
plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, color='darkorange', lw=2,
         label='ROC curve (AUC = %0.3f)' % roc_auc)
```

```
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve - Random Forest Classifier')
plt.legend(loc="lower right")
plt.show()
```



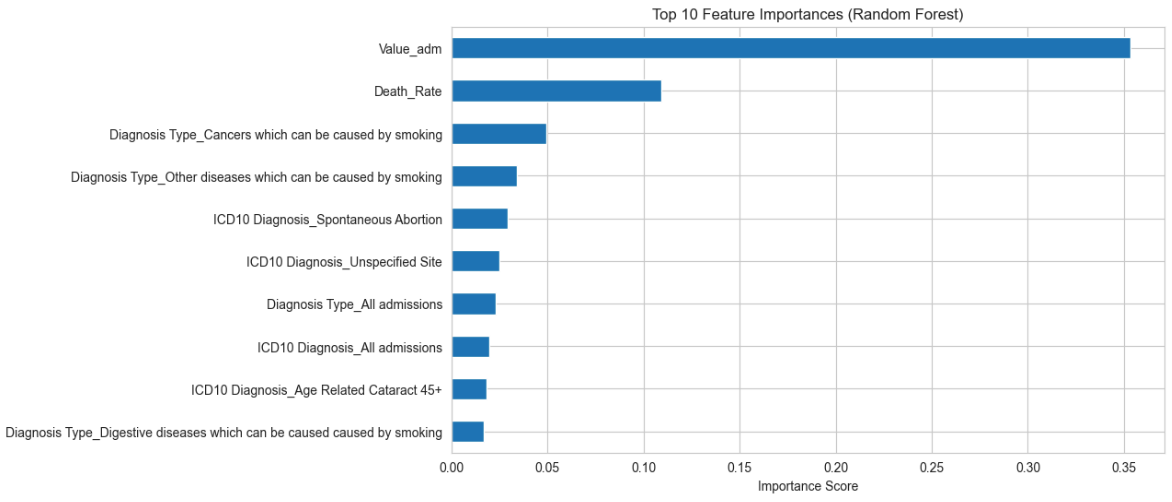
In [72]:
```
# Plot actual vs predicted for Random Forest
plt.figure(figsize=(8,6))
plt.scatter(y_test, y_pred_rf, alpha=0.7)
plt.xlabel("Actual Fatalities")
plt.ylabel("Predicted Fatalities")
plt.title("Random Forest: Actual vs Predicted Fatalities")
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'r--')
plt.show()
```



model interpretation

In [76]:
```
import pandas as pd
import matplotlib.pyplot as plt

# Get feature importances from your fitted RF classifier
importances = rf_clf.feature_importances_

# Put into DataFrame for readability
feat_importances = pd.Series(importances, index=X_train.columns)
feat_importances = feat_importances.sort_values(ascending=False)

print("Top 10 Important Features:")
print(feat_importances.head(10))

# Plot
plt.figure(figsize=(10, 6))
feat_importances.head(10).plot(kind='barh')
plt.gca().invert_yaxis()
plt.title('Top 10 Feature Importances (Random Forest)')
plt.xlabel('Importance Score')
plt.show()
```

```
Top 10 Important Features:
Value_adm                                                             0.3532
97
Death_Rate                                                            0.1092
62
Diagnosis Type_Cancers which can be caused by smoking                 0.0495
78
Diagnosis Type_Other diseases which can be caused by smoking          0.0339
21
ICD10 Diagnosis_Spontaneous Abortion                                  0.0291
29
ICD10 Diagnosis_Unspecified Site                                      0.0252
33
Diagnosis Type_All admissions                                         0.0231
11
ICD10 Diagnosis_All admissions                                        0.0198
72
ICD10 Diagnosis_Age Related Cataract 45+                              0.0184
54
Diagnosis Type_Digestive diseases which can be caused caused by smoking    0.0171
05
dtype: float64
```



In [ ]:

Deeployment

In [6]:
```python
import pandas as pd
import numpy as np
import joblib
from sklearn.ensemble import RandomForestRegressor, RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

df = pd.read_csv("merged_featured.csv")

df.columns = df.columns.str.replace(r'\s+', ' ', regex=True).str.strip()

df = df.dropna(subset=['Death_Rate']).copy()

df_class = df.dropna(subset=['Value_fat']).copy()
threshold = df_class['Value_fat'].median()
df_class['High_Fatality'] = np.where(df_class['Value_fat'] >= threshold, 1, 0)
```

```
# Now the features match:
features = [
    'Smoking Prevalence',
    'Tobacco Price Index',
    'Retail Prices Index',
    "Real Households' Disposable Income",
    'SmokingPrice_Interaction',
    'Sex_Male',
    'Policy_Era_Pre-2010',
    'ICD10 Diagnosis',
    'Diagnosis Type'
]
X_reg = df[features].copy()
y_reg = df['Death_Rate']


# --- REGRESSION ---
X_reg = df[features].copy()
y_reg = df['Death_Rate']

# --- CLASSIFICATION ---
X_clf = df_class[features].copy()
y_clf = df_class['High_Fatality']

# One-hot encode categorical columns
X_reg = pd.get_dummies(X_reg, columns=['ICD10 Diagnosis', 'Diagnosis Type'])
X_clf = pd.get_dummies(X_clf, columns=['ICD10 Diagnosis', 'Diagnosis Type'])

# Align columns
feature_order = X_reg.columns.tolist()

# Save feature order
joblib.dump(feature_order, 'feature_order.pkl')

# Scale
scaler = StandardScaler()
X_reg_scaled = scaler.fit_transform(X_reg)
X_clf_scaled = scaler.transform(X_clf)

joblib.dump(scaler, 'scaler.pkl')

# Train models
regressor = RandomForestRegressor(random_state=42)
regressor.fit(X_reg_scaled, y_reg)

classifier = RandomForestClassifier(random_state=42)
classifier.fit(X_clf_scaled, y_clf)

# Save
joblib.dump(regressor, 'regressor.pkl')
joblib.dump(classifier, 'classifier.pkl')

print(" Models trained and saved as regressor.pkl and classifier.pkl.")
```

```
Models trained and saved as regressor.pkl and classifier.pkl.
```

```
In [ ]:
```

PROJECT SUMMARY

Processes & Results

Data Preparation

- **Scope:** Tobacco use, economic factors, ICD10 diagnosis, mortality (`Death_Rate` and `Value_fat`) from **2004–2015**.
- Removed missing values for target variables.
- Cleaned hidden characters from column names (`\r` , `\n`).
- Encoded diagnosis fields, scaled numerical data.
- Engineered a binary `High_Fatality` flag for classification.

Exploratory Data Analysis (EDA)

- Confirmed trends:
  - Tobacco price index generally rose over time.
  - Smoking prevalence gradually decreased.
  - Household disposable income fluctuated slightly.
- Noted relationships:
  - Higher tobacco prices tend to align with lower smoking prevalence.
  - Specific diagnoses (cancers, circulatory diseases) showed higher `Death_Rate` .

Feature Engineering

- Added `SmokingPrice_Interaction` to combine smoking prevalence and price effect.
- Binarized `Value_fat` into `High_Fatality` class using median split.

Model Building

✅ **Regression:**

- **Linear Regression**
  - RMSE: **3967.65**
  - R^2: **0.9971**
- **Random Forest Regressor**
  - RMSE: **2466.61**
  - R^2: **0.9989**

✅ **Classification:**

- **Random Forest Classifier**
  - **Accuracy: 0.98**
  - **Confusion Matrix:**

    ```
    [[67  0]
     [ 3 80]]
    ```
  - **Classification Report:**

    | Metric | Class 0 | Class 1 | | --------- | ------- | ------- | | Precision | 0.96 | 1.00 | | Recall | 1.00 | 0.96 | | F1-Score | 0.98 | 0.98 |
  - **ROC AUC: 0.9993**

Deployment

- Built **Streamlit web app** for live prediction:Check here:
  - streamlit could not be deployed on jupyter notebook
  - Mode switch: Regression (`Death_Rate`) vs Classification (`High_Fatality`).
  - User inputs: policy, income, prices, diagnosis, smoking rates.
  - SHAP explanations show how each feature influences prediction.

Key Insights

- Strong prediction power for both tasks:
  - **Very high R^2** and low RMSE → reliable mortality rate estimates.
  - **Excellent classifier performance** → clear separation of high vs low fatality risk.
- Tobacco pricing and smoking prevalence remain **key policy levers** to reduce mortality.
- Diagnosis type and interaction of price + prevalence have strong impact on outcomes.

Final Conclusion

The project **successfully combines EDA, feature engineering, ML modeling, and deployment**:

- Predicts `Death_Rate` with high accuracy.
- Classifies diagnoses by high vs low fatality risk with near-perfect ROC AUC.
- Provides a practical **Streamlit dashboard** for public health decision-makers and researchers.

- Demonstrates the power of **data-driven policy analysis** for tobacco-related mortality control.

🚀 Performance Summary Table

Model	RMSE	R^2	Accuracy	ROC AUC
Linear Regression	3967.65	0.9971	—	—
Random Forest Regression	2466.61	0.9989	—	—
Random Forest Classification	—	—	98%	0.9993

```
In [ ]:
```