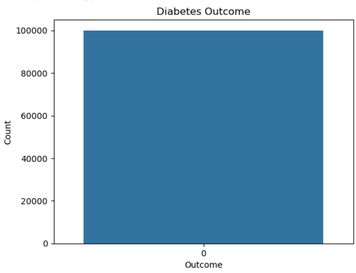
In [5]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, confusion_matrix, classification_report

# Load dataset
# Replace 'diabetes_prediction_dataset.csv' with the actual path to your dataset
data = pd.read_csv('diabetes_prediction_dataset.csv')

# Data Exploration
print(data.head())
print(data['diabetes'].value_counts())

# Data Visualization
sns.countplot(data['diabetes'])
plt.title('Diabetes Outcome')
plt.xlabel('Outcome')
plt.ylabel('Count')
plt.show()

# Data Preprocessing
X = data.drop('diabetes', axis=1)
y = data['diabetes']

# Feature Scaling
scaler = StandardScaler()
X = scaler.fit_transform(X)

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

# Model Building
models = {
    'Random Forest': RandomForestClassifier(),
    'KNN': KNeighborsClassifier(),
    'AdaBoost': AdaBoostClassifier(),
    'Decision Tree': DecisionTreeClassifier()
}

results = {}

for name, model in models.items():
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred)
    precision = precision_score(y_test, y_pred)
    recall = recall_score(y_test, y_pred)
    f1 = f1_score(y_test, y_pred)
    cm = confusion_matrix(y_test, y_pred)
    results[name] = {
        'Accuracy': accuracy,
        'Precision': precision,
        'Recall': recall,
        'F1-score': f1,
        'Confusion Matrix': cm
    }

    # Results and Visualizations
    print(f"Model: {name}")
    for metric, value in results[name].items():
        print(f"{metric}: {value}")
    print("\n")

    # Confusion Matrix Visualization
    plt.figure()
    sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
    plt.xlabel('Predicted')
    plt.ylabel('Actual')
    plt.title(f'Confusion Matrix - {name}')
    plt.show()

# Linear Regression does not have a confusion matrix, so it's handled separately
linear_model = LinearRegression()
linear_model.fit(X_train, y_train)
y_pred = linear_model.predict(X_test)
rmse = np.sqrt(((y_pred - y_test) ** 2).mean())
print(f"Linear Regression RMSE: {rmse}")

# ATS-Friendly Classification Report
class_report = classification_report(y_test, y_pred.round(), output_dict=True)
ats_friendly_report = """
ATS-friendly Classification Report:
    - Not Diabetic Class:
        Precision: {:.2f}
        Recall: {:.2f}
        F1-score: {:.2f}
        Support: {}

    - Diabetic Class:
        Precision: {:.2f}
        Recall: {:.2f}
        F1-score: {:.2f}
        Support: {}
""".format(
    class_report['0']['precision'],
    class_report['0']['recall'],
    class_report['0']['f1-score'],
    class_report['0']['support'],
    class_report['1']['precision'],
    class_report['1']['recall'],
    class_report['1']['f1-score'],
    class_report['1']['support']
)

print(ats_friendly_report)
```

```
   gender   age  hypertension  heart_disease smoking_history    bmi  \
0  Female  80.0             0              1           never  25.19
1  Female  54.0             0              0        No Info  27.32
2    Male  28.0             0              0           never  27.32
3  Female  36.0             0              0         current  23.45
4    Male  76.0             1              1         current  20.14

   HbA1c_level  blood_glucose_level  diabetes
0          6.6                  140         0
1          6.6                   80         0
2          5.7                  158         0
3          5.0                  155         0
4          4.8                  155         0
0    91500
1     8500
Name: diabetes, dtype: int64
```




```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[5], line 34
     32 # Feature Scaling
     33 scaler = StandardScaler()
---> 34 X = scaler.fit_transform(X)
     36 # Data Splitting
     37 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

File ~\AppData\Roaming\Python\Python310\site-packages\sklearn\utils\_set_output.py:295, in _wrap_method_output.<locals>.wrapped(self, X, *args, **kwargs)
    293 @wraps(f)
    294 def wrapped(self, X, *args, **kwargs):
--> 295     data_to_wrap = f(self, X, *args, **kwargs)
    296     if isinstance(data_to_wrap, tuple):
    297         # only wrap the first output for cross decomposition
    298         return_tuple = (
    299             _wrap_data_with_container(method, data_to_wrap[0], X, self),
    300             *data_to_wrap[1:],
    301         )

File ~\AppData\Roaming\Python\Python310\site-packages\sklearn\base.py:1098, in TransformerMixin.fit_transform(self, X, y, **fit_params)
   1083         warnings.warn(
   1084             (
   1085                 f"This object ({self.__class__.__name__}) has a `transform`"
   (...)
   1093                 UserWarning,
   1094             )
   1096 if y is None:
   1097     # fit method of arity 1 (unsupervised transformation)
-> 1098     return self.fit(X, **fit_params).transform(X)
   1099 else:
   1100     # fit method of arity 2 (supervised transformation)
   1101     return self.fit(X, y, **fit_params).transform(X)

File ~\AppData\Roaming\Python\Python310\site-packages\sklearn\preprocessing\_data.py:876, in StandardScaler.fit(self, X, y, sample_weight)
    874 # Reset internal state before fitting
    875 self._reset()
--> 876 return self.partial_fit(X, y, sample_weight)

File ~\AppData\Roaming\Python\Python310\site-packages\sklearn\base.py:1474, in _fit_context.<locals>.decorator.<locals>.wrapper(estimator, *args, **kwargs)
   1467     estimator._validate_params()
   1469 with config_context(
   1470     skip_parameter_validation=(
   1471         prefer_skip_nested_validation or global_skip_validation
   1472     )
   1473 ):
-> 1474     return fit_method(estimator, *args, **kwargs)

File ~\AppData\Roaming\Python\Python310\site-packages\sklearn\preprocessing\_data.py:912, in StandardScaler.partial_fit(self, X, y, sample_weight)
    880 """Online computation of mean and std on X for later scaling.
    881
    882 All of X is processed as a single batch. This is intended for cases
    (...)
    909     Fitted scaler.
    910 """
    911 first_call = not hasattr(self, "n_samples_seen_")
--> 912 X = self._validate_data(
    913     X,
    914     accept_sparse=("csr", "csc"),
    915     dtype=FLOAT_DTYPES,
    916     force_all_finite="allow-nan",
    917     reset=first_call,
    918 )
    919 n_features = X.shape[1]
    921 if sample_weight is not None:

File ~\AppData\Roaming\Python\Python310\site-packages\sklearn\base.py:633, in BaseEstimator._validate_data(self, X, y, reset, validate_separately, cast_to_ndarray, **check_params)
    631         out = X, y
    632 elif not no_val_X and no_val_y:
--> 633     out = check_array(X, input_name="X", **check_params)
    634 elif no_val_X and not no_val_y:
    635     out = _check_y(y, **check_params)

File ~\AppData\Roaming\Python\Python310\site-packages\sklearn\utils\validation.py:997, in check_array(array, accept_sparse, accept_large_sparse, dtype, order, copy, force_all_finite, ensure_2d, allow_nd, ensure_min_samples, ensure_min_features, estimator, input_name)
    995         array = xp.astype(array, dtype, copy=False)
    996     else:
--> 997         array = _asarray_with_order(array, order=order, dtype=dtype, xp=xp)
    998 except ComplexWarning as complex_warning:
    999     raise ValueError(
   1000         "Complex data not supported\n{}\n".format(array)
   1001     ) from complex_warning

File ~\AppData\Roaming\Python\Python310\site-packages\sklearn\utils\_array_api.py:521, in _asarray_with_order(array, dtype, order, copy, xp)
    519         array = numpy.array(array, order=order, dtype=dtype)
    520     else:
--> 521         array = numpy.asarray(array, order=order, dtype=dtype)
    523     # At this point array is a NumPy ndarray. We convert it to an array
    524     # container that is consistent with the input's namespace.
    525     return xp.asarray(array)

File C:\ProgramData\anaconda3\lib\site-packages\pandas\core\generic.py:2070, in NDFrame.__array__(self, dtype)
   2069 def __array__(self, dtype: npt.DTypeLike | None = None) -> np.ndarray:
-> 2070     return np.asarray(self._values, dtype=dtype)

ValueError: could not convert string to float: 'Female'
```

In [ ]: