In [1]:

```
#### Step 1: Import Required Libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, classification_report, confusion_m
```

In [2]:

```
#### Step 2: Load and Inspect the Data
df = pd.read_csv("bigml_59c28831336c6604c800002a.csv")
print(df.head())
print(df.info())
print(df.describe())
```

```
  state  account length  area code phone number international plan  \
0    KS             128        415     382-4657                 no
1    OH             107        415     371-7191                 no
2    NJ             137        415     358-1921                 no
3    OH              84        408     375-9999                yes
4    OK              75        415     330-6626                yes

  voice mail plan  number vmail messages  total day minutes  total day cal
ls  \
0             yes                     25              265.1              1
10
1             yes                     26              161.6              1
23
2              no                      0              243.4              1
14
3              no                      0              299.4
71
4              no                      0              166.7              1
13
```

In [3]:

```
#### Step 3: Check for Missing Values
print(df.isnull().sum())
```

```
state                     0
account length            0
area code                 0
phone number              0
international plan        0
voice mail plan           0
number vmail messages     0
total day minutes         0
total day calls           0
total day charge          0
total eve minutes         0
total eve calls           0
total eve charge          0
total night minutes       0
total night calls         0
total night charge        0
total intl minutes        0
total intl calls          0
total intl charge         0
customer service calls    0
churn                     0
dtype: int64
```

In [4]:

```
#### Step 4: Data Preprocessing
# Drop unnecessary columns
df.drop(columns=['phone number'], inplace=True)

# Convert target column to numerical
df['churn'] = df['churn'].astype(int)

# Convert categorical columns to numerical
df['international plan'] = df['international plan'].map({'yes': 1, 'no': 0})
df['voice mail plan'] = df['voice mail plan'].map({'yes': 1, 'no': 0})

# One-hot encoding for 'state'
df = pd.get_dummies(df, columns=['state'], drop_first=True)
```

In [5]:

```
#### Step 5: Define Features and Target
y = df['churn']
X = df.drop(columns=['churn'])
```

In [6]:

```
#### Step 6: Split Data into Training and Testing Sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, randc
```

In [7]:

```
#### Step 7: Scale the Features
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

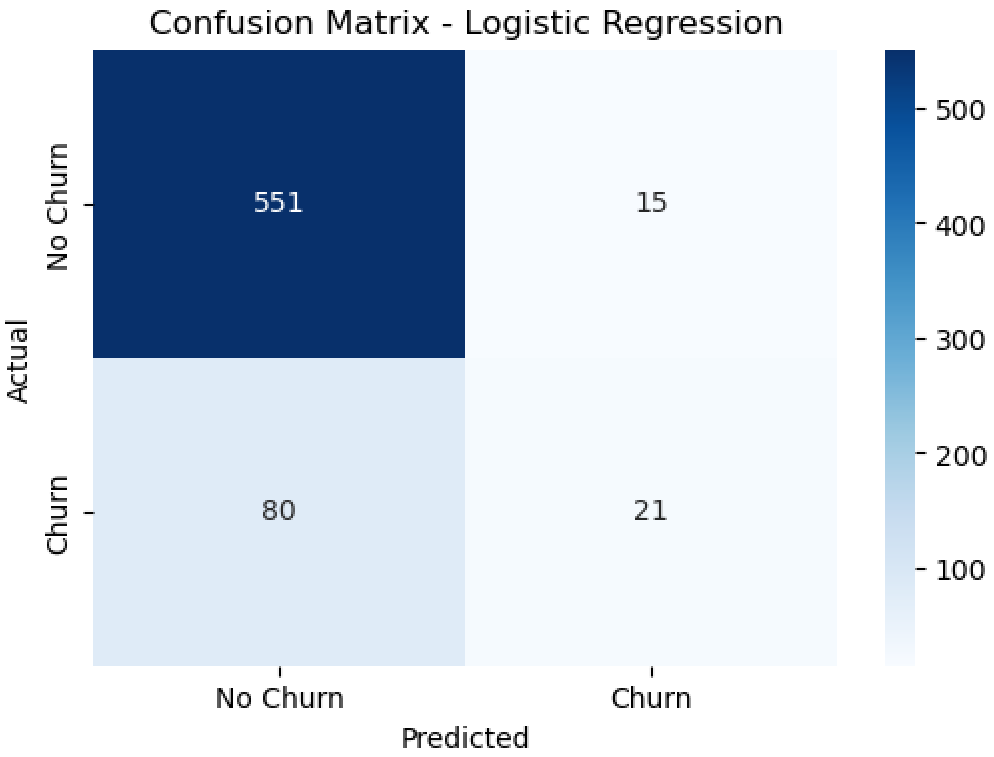
In [8]:

```
#### Step 8: Train Logistic Regression Model
log_reg = LogisticRegression()
log_reg.fit(X_train_scaled, y_train)
y_pred_log = log_reg.predict(X_test_scaled)
```

In [9]:

```
#### Step 9: Evaluate Logistic Regression Model
print("Logistic Regression Accuracy:", accuracy_score(y_test, y_pred_log))
print(classification_report(y_test, y_pred_log))

# Confusion Matrix
plt.figure(figsize=(6, 4))
sns.heatmap(confusion_matrix(y_test, y_pred_log), annot=True, fmt='d', cmap="B
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix - Logistic Regression")
plt.show()
```

```
Logistic Regression Accuracy: 0.8575712143928036
              precision    recall  f1-score   support

           0       0.87      0.97      0.92       566
           1       0.58      0.21      0.31       101

    accuracy                           0.86       667
   macro avg       0.73      0.59      0.61       667
weighted avg       0.83      0.86      0.83       667
```

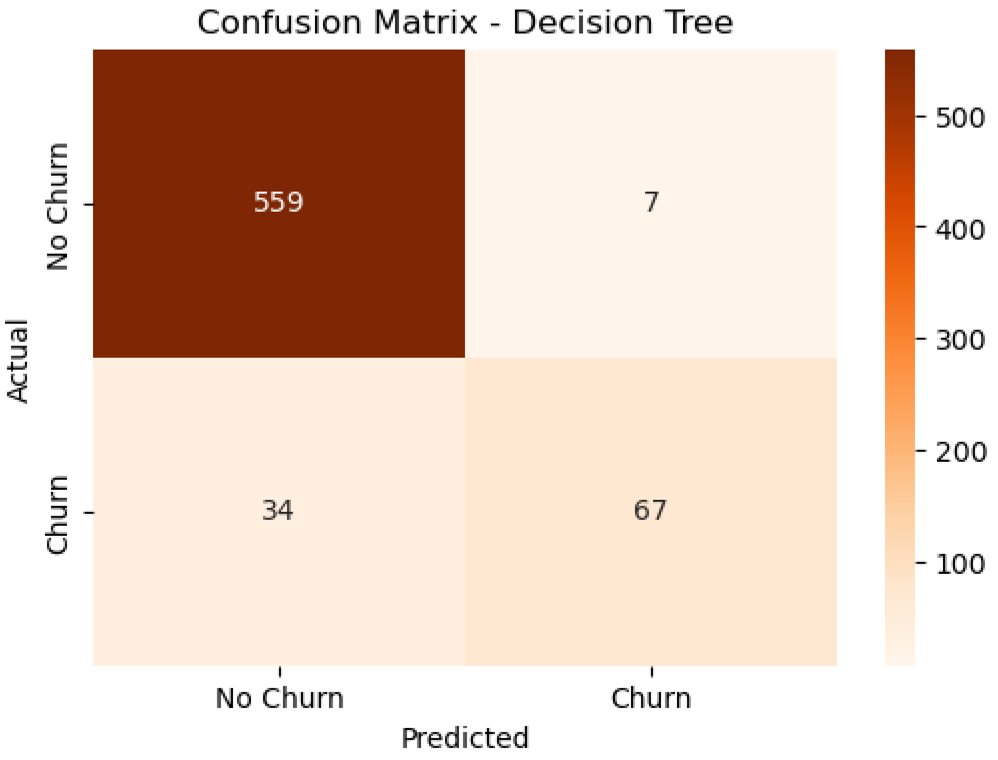


In [10]:

```
#### Step 10: Train Decision Tree Model
decision_tree = DecisionTreeClassifier(max_depth=5, random_state=42)
decision_tree.fit(X_train, y_train)
y_pred_tree = decision_tree.predict(X_test)
```

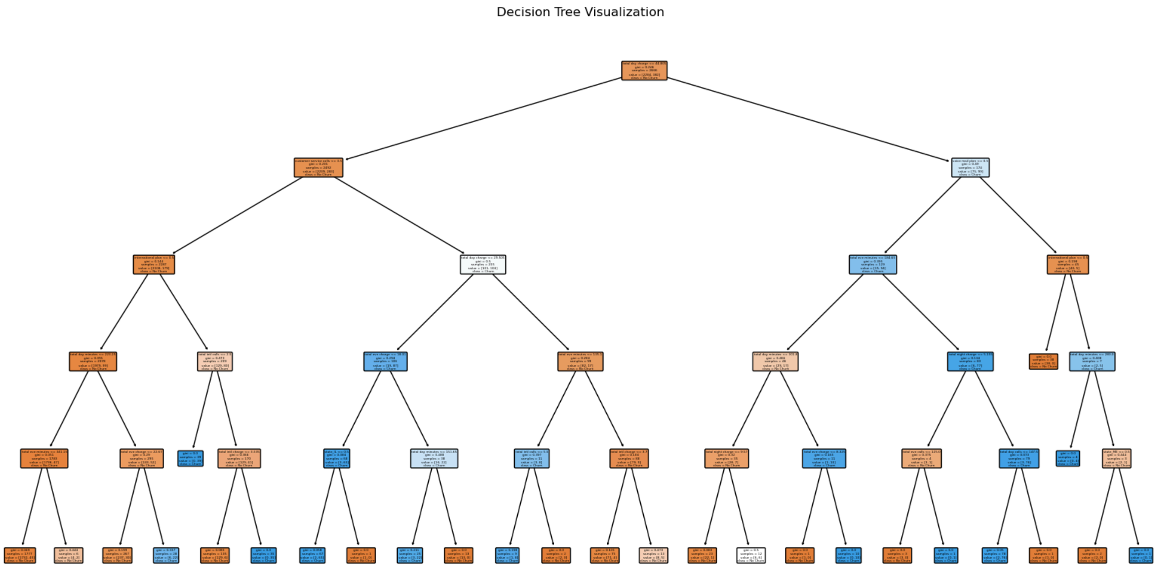
In [11]:
```
#### Sstep 11: Evaluate Decision Tree Model
print("Decision Tree Accuracy:", accuracy_score(y_test, y_pred_tree))
print(classification_report(y_test, y_pred_tree))

# Confusion Matrix for Decision Tree
plt.figure(figsize=(6, 4))
sns.heatmap(confusion_matrix(y_test, y_pred_tree), annot=True, fmt='d', cmap="
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix - Decision Tree")
plt.show()
```

```
Decision Tree Accuracy: 0.9385307346326837
              precision    recall  f1-score   support

           0       0.94      0.99      0.96       566
           1       0.91      0.66      0.77       101

    accuracy                           0.94       667
   macro avg       0.92      0.83      0.87       667
weighted avg       0.94      0.94      0.93       667
```

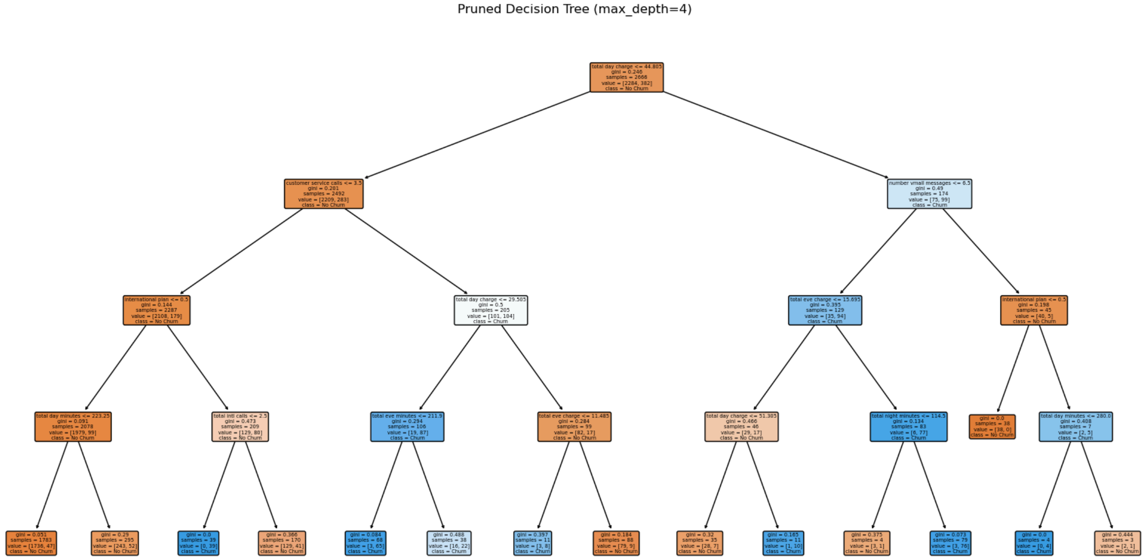


In [13]:

```
from sklearn.tree import plot_tree

plt.figure(figsize=(20, 10))
plot_tree(decision_tree, feature_names=X.columns, class_names=['No Churn', 'Ch
plt.title("Decision Tree Visualization")
plt.show()
```



In [14]:
```
# Train a new Decision Tree with a depth limit
decision_tree = DecisionTreeClassifier(max_depth=4, random_state=42)
decision_tree.fit(X_train, y_train)

# Plot the pruned tree
plt.figure(figsize=(20, 10))
plot_tree(decision_tree, feature_names=X.columns, class_names=['No Churn', 'Ch
plt.title("Pruned Decision Tree (max_depth=4)")
plt.show()
```


Pruned Decision Tree (max_depth=4)
total day charge <= 44.805
gini = 0.246
samples = 2666
value = [2284, 382]
class = No Churn
customer service calls <= 3.5
gini = 0.201
samples = 2492
value = [2209, 283]
class = No Churn
number vmail messages <= 6.5
gini = 0.49
samples = 174
value = [75, 99]
class = Churn
international plan <= 0.5
gini = 0.144
samples = 2287
value = [2108, 179]
class = No Churn
total day charge <= 29.505
gini = 0.5
samples = 205
value = [101, 104]
class = Churn
total eve charge <= 15.695
gini = 0.395
samples = 129
value = [35, 94]
class = Churn
international plan <= 0.5
gini = 0.198
samples = 45
value = [40, 5]
class = No Churn
total day minutes <= 223.25
gini = 0.091
samples = 2078
value = [1979, 99]
class = No Churn
total intl calls <= 2.5
gini = 0.473
samples = 209
value = [129, 80]
class = No Churn
total eve minutes <= 211.9
gini = 0.294
samples = 106
value = [19, 87]
class = Churn
total eve charge <= 11.485
gini = 0.284
samples = 99
value = [82, 17]
class = No Churn
total day charge <= 51.305
gini = 0.466
samples = 46
value = [29, 17]
class = No Churn
total night minutes <= 114.5
gini = 0.134
samples = 83
value = [6, 77]
class = Churn
gini = 0.0
samples = 38
value = [38, 0]
class = No Churn
total day minutes <= 280.0
gini = 0.408
samples = 7
value = [2, 5]
class = Churn
gini = 0.051
samples = 1783
value = [1736, 47]
class = No Churn
gini = 0.29
samples = 295
value = [243, 52]
class = No Churn
gini = 0.0
samples = 39
value = [0, 39]
class = Churn
gini = 0.366
samples = 170
value = [129, 41]
class = No Churn
gini = 0.084
samples = 68
value = [3, 65]
class = Churn
gini = 0.488
samples = 38
value = [16, 22]
class = Churn
gini = 0.397
samples = 11
value = [3, 8]
class = Churn
gini = 0.184
samples = 88
value = [79, 9]
class = No Churn
gini = 0.32
samples = 35
value = [28, 7]
class = No Churn
gini = 0.165
samples = 11
value = [1, 10]
class = Churn
gini = 0.375
samples = 4
value = [3, 1]
class = No Churn
gini = 0.073
samples = 79
value = [3, 76]
class = Churn
gini = 0.0
samples = 4
value = [0, 4]
class = Churn
gini = 0.444
samples = 3
value = [2, 1]
class = No Churn