assignment8

September 24, 2024

[1]:
```
sc
```

[1]:
```
<SparkContext master=local[*] appName=PySparkShell>
```

[2]:
```
spark
```

[2]:
```
<pyspark.sql.session.SparkSession at 0x7fe1936373c8>
```

[3]:
```
import matplotlib.pyplot as plt
import seaborn as sns
from pyspark.sql.functions import count, when, isnull
```

[106]:
```
churn_data = spark.read.csv("file:///home/hadoop/Downloads/Telco_Customer_Churn.
↪csv",
                            inferSchema=True, header=True)
churn_data.show()
```

```
+----------+------+-------------+-------+----------+------+------------+--------
--------+---------------+-------------------+-------------------+---------------
----+-------------------+-------------------+-------------------+--------------+
----------------+--------------------+--------------+------------+-----+
|customerID|gender|SeniorCitizen|Partner|Dependents|tenure|PhoneService|
MultipleLines|InternetService|     OnlineSecurity|       OnlineBackup|
DeviceProtection|         TechSupport|        StreamingTV|    StreamingMovies|
Contract|PaperlessBilling|
PaymentMethod|MonthlyCharges|TotalCharges|Churn|
+----------+------+-------------+-------+----------+------+------------+--------
--------+---------------+-------------------+-------------------+---------------
----+-------------------+-------------------+-------------------+--------------+
----------------+--------------------+--------------+------------+-----+
|7590-VHVEG|Female|            0|    Yes|        No|     1|          No|No phone
service|            DSL|                 No|                Yes|
No|                 No|                 No|                 No|Month-to-month|
Yes|    Electronic check|         29.85|       29.85|   No|
|5575-GNVDE|  Male|            0|     No|        No|    34|         Yes|
No|           DSL|                Yes|                 No|                Yes|
No|                 No|                 No|      One year|              No|
Mailed check|          56.95|      1889.5|   No|
```

```
|3668-QPYBK|  Male|            0|     No|         No|     2|         Yes|
No|           DSL|                Yes|                Yes|                No|
No|                No|                 No|Month-to-month|            Yes|
Mailed check|         53.85|      108.15|  Yes|
|7795-CFOCW|  Male|            0|     No|         No|    45|          No|No phone
service|           DSL|                Yes|                No|
Yes|               Yes|                No|                 No|      One year|
No|Bank transfer (au…|          42.3|     1840.75|   No|
|9237-HQITU|Female|            0|     No|         No|     2|         Yes|
No|    Fiber optic|                 No|                 No|                No|
No|                No|                 No|Month-to-month|            Yes|
Electronic check|          70.7|       151.65|  Yes|
|9305-CDSKC|Female|            0|     No|         No|     8|         Yes|
Yes|    Fiber optic|                 No|                 No|               Yes|
No|               Yes|                Yes|Month-to-month|            Yes|
Electronic check|         99.65|       820.5|  Yes|
|1452-KIOVK|  Male|            0|     No|        Yes|    22|         Yes|
Yes|    Fiber optic|                 No|                Yes|                No|
No|               Yes|                 No|Month-to-month|
Yes|Credit card (auto…|          89.1|      1949.4|   No|
|6713-OKOMC|Female|            0|     No|         No|    10|          No|No phone
service|           DSL|                Yes|                No|
No|                No|                 No|                No|Month-to-month|
No|        Mailed check|         29.75|       301.9|   No|
|7892-POOKP|Female|            0|    Yes|         No|    28|         Yes|
Yes|    Fiber optic|                 No|                 No|               Yes|
Yes|               Yes|                Yes|Month-to-month|            Yes|
Electronic check|         104.8|     3046.05|  Yes|
|6388-TABGU|  Male|            0|     No|        Yes|    62|         Yes|
No|           DSL|                Yes|                Yes|                No|
No|                No|                 No|      One year|             No|Bank
transfer (au…|         56.15|     3487.95|   No|
|9763-GRSKD|  Male|            0|    Yes|        Yes|    13|         Yes|
No|           DSL|                Yes|                 No|                No|
No|                No|                 No|Month-to-month|            Yes|
Mailed check|         49.95|      587.45|   No|
|7469-LKBCI|  Male|            0|     No|         No|    16|         Yes|
No|              No|No internet service|No internet service|No internet
service|No internet service|No internet service|No internet service|      Two
year|              No|Credit card (auto…|         18.95|       326.8|   No|
|8091-TTVAX|  Male|            0|    Yes|         No|    58|         Yes|
Yes|    Fiber optic|                 No|                 No|               Yes|
No|               Yes|                Yes|      One year|
No|Credit card (auto…|        100.35|      5681.1|   No|
|0280-XJGEX|  Male|            0|     No|         No|    49|         Yes|
Yes|    Fiber optic|                 No|                Yes|               Yes|
No|               Yes|                Yes|Month-to-month|            Yes|Bank
transfer (au…|         103.7|      5036.3|  Yes|
```

```
|5129-JLPIS|  Male|            0|     No|        No|    25|         Yes|
No|    Fiber optic|               Yes|                 No|                Yes|
Yes|                Yes|                Yes|Month-to-month|            Yes|
Electronic check|         105.5|     2686.05|   No|
|3655-SNQYZ|Female|            0|    Yes|       Yes|    69|         Yes|
Yes|    Fiber optic|               Yes|                Yes|                Yes|
Yes|                Yes|                Yes|      Two year|
No|Credit card (auto…|        113.25|     7895.15|   No|
|8191-XWSZG|Female|            0|     No|        No|    52|         Yes|
No|               No|No internet service|No internet service|No internet
service|No internet service|No internet service|No internet service|      One
year|               No|        Mailed check|         20.65|     1022.95|   No|
|9959-WOFKT|  Male|            0|     No|       Yes|    71|         Yes|
Yes|    Fiber optic|               Yes|                 No|                Yes|
No|                 Yes|                Yes|      Two year|              No|Bank
transfer (au…|         106.7|     7382.25|   No|
|4190-MFLUW|Female|            0|    Yes|       Yes|    10|         Yes|
No|             DSL|               No|                 No|                Yes|
Yes|                 No|                 No|Month-to-month|
No|Credit card (auto…|          55.2|      528.35|  Yes|
|4183-MYFRB|Female|            0|     No|        No|    21|         Yes|
No|    Fiber optic|                No|                Yes|                Yes|
No|                 No|                Yes|Month-to-month|            Yes|
Electronic check|         90.05|      1862.9|   No|
+----------+------+-------------+-------+----------+------+------------+--------
--------+---------------+-------------------+-------------------+---------------
----+-------------------+-------------------+-------------------+--------------+
----------------+-------------------+-------------+------------+-----+
only showing top 20 rows
```

[5]:
```python
churn_data.printSchema()
```

```
root
 |-- customerID: string (nullable = true)
 |-- gender: string (nullable = true)
 |-- SeniorCitizen: integer (nullable = true)
 |-- Partner: string (nullable = true)
 |-- Dependents: string (nullable = true)
 |-- tenure: integer (nullable = true)
 |-- PhoneService: string (nullable = true)
 |-- MultipleLines: string (nullable = true)
 |-- InternetService: string (nullable = true)
 |-- OnlineSecurity: string (nullable = true)
 |-- OnlineBackup: string (nullable = true)
 |-- DeviceProtection: string (nullable = true)
 |-- TechSupport: string (nullable = true)
 |-- StreamingTV: string (nullable = true)
```

```
 |-- StreamingMovies: string (nullable = true)
 |-- Contract: string (nullable = true)
 |-- PaperlessBilling: string (nullable = true)
 |-- PaymentMethod: string (nullable = true)
 |-- MonthlyCharges: double (nullable = true)
 |-- TotalCharges: string (nullable = true)
 |-- Churn: string (nullable = true)
```

[107]:
```
from pyspark.sql.types import FloatType

# changing datatype of TotalCharges column from String to Float
churn_data = churn_data.withColumn("TotalCharges", churn_data["TotalCharges"].
↪cast(FloatType()))
```

[7]:
```
# checking for null values
churn_data.select([count(when(isnull(col), col)).alias(col) for col in␣
↪churn_data.columns]).collect()
```

[7]:
```
[Row(customerID=0, gender=0, SeniorCitizen=0, Partner=0, Dependents=0, tenure=0,
PhoneService=0, MultipleLines=0, InternetService=0, OnlineSecurity=0,
OnlineBackup=0, DeviceProtection=0, TechSupport=0, StreamingTV=0,
StreamingMovies=0, Contract=0, PaperlessBilling=0, PaymentMethod=0,
MonthlyCharges=0, TotalCharges=11, Churn=0)]
```

[108]:
```
churn_data = churn_data.dropna()
```

[9]:
```
churn_data.createOrReplaceTempView('churnData')
```

0.0.1 a) Analyze how customer retention varies based on how long the customer has stayed with the company (tenure).

[93]:
```
resultQuery = spark.sql("""
    SELECT tenure, round(AVG(CASE WHEN Churn="No" THEN 1 ELSE 0 END)*100, 3) AS␣
↪retention_rate,
                   SUM(1) AS total
    FROM churnData
    -- WHERE tenure > 0
    GROUP BY tenure
    ORDER BY tenure
""")
resultQuery.show(75)

# INSIGHT: we see that as tenure increases, there is a reduction in the
# number of people who left the company when compared to total customers with␣
↪same tenure
```

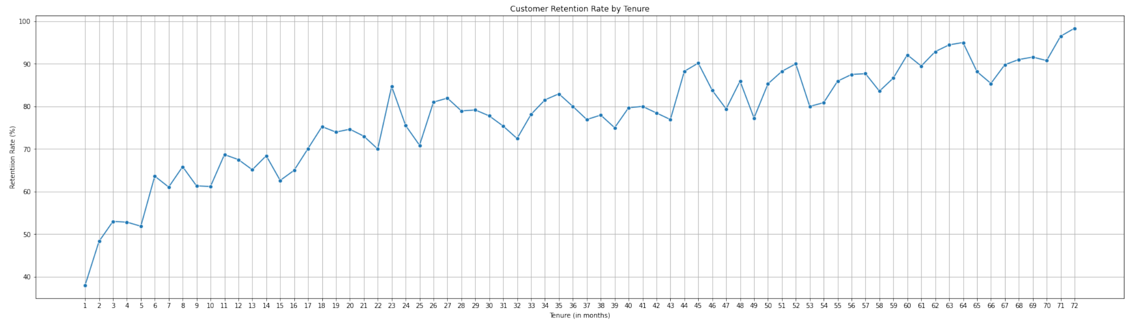
```
# thus tenure and number of customers who left have an inverse relation
```

```
+------+--------------+-----+
|tenure|retention_rate|total|
+------+--------------+-----+
|     1|         38.01|  613|
|     2|        48.319|  238|
|     3|          53.0|  200|
|     4|        52.841|  176|
|     5|         51.88|  133|
|     6|        63.636|  110|
|     7|        61.069|  131|
|     8|        65.854|  123|
|     9|        61.345|  119|
|    10|        61.207|  116|
|    11|        68.687|   99|
|    12|        67.521|  117|
|    13|        65.138|  109|
|    14|        68.421|   76|
|    15|        62.626|   99|
|    16|          65.0|   80|
|    17|        70.115|   87|
|    18|        75.258|   97|
|    19|        73.973|   73|
|    20|        74.648|   71|
|    21|        73.016|   63|
|    22|          70.0|   90|
|    23|        84.706|   85|
|    24|        75.532|   94|
|    25|        70.886|   79|
|    26|        81.013|   79|
|    27|        81.944|   72|
|    28|        78.947|   57|
|    29|        79.167|   72|
|    30|        77.778|   72|
|    31|        75.385|   65|
|    32|        72.464|   69|
|    33|        78.125|   64|
|    34|        81.538|   65|
|    35|        82.955|   88|
|    36|          80.0|   50|
|    37|        76.923|   65|
|    38|        77.966|   59|
|    39|          75.0|   56|
|    40|        79.688|   64|
|    41|          80.0|   70|
|    42|        78.462|   65|
```

```
|    43|        76.923|   65|
|    44|        88.235|   51|
|    45|        90.164|   61|
|    46|        83.784|   74|
|    47|        79.412|   68|
|    48|        85.938|   64|
|    49|        77.273|   66|
|    50|        85.294|   68|
|    51|        88.235|   68|
|    52|          90.0|   80|
|    53|          80.0|   70|
|    54|        80.882|   68|
|    55|        85.938|   64|
|    56|          87.5|   80|
|    57|        87.692|   65|
|    58|        83.582|   67|
|    59|        86.667|   60|
|    60|        92.105|   76|
|    61|        89.474|   76|
|    62|        92.857|   70|
|    63|        94.444|   72|
|    64|          95.0|   80|
|    65|        88.158|   76|
|    66|        85.393|   89|
|    67|        89.796|   98|
|    68|          91.0|  100|
|    69|        91.579|   95|
|    70|        90.756|  119|
|    71|        96.471|  170|
|    72|        98.343|  362|
+------+--------------+-----+
```

[11]:

```
resultDF = resultQuery.toPandas()

plt.figure(figsize=(30, 8))
sns.lineplot(data=resultDF, x='tenure', y='retention_rate', marker='o')
plt.title('Customer Retention Rate by Tenure')
plt.xlabel('Tenure (in months)')
plt.ylabel('Retentiion Rate (%)')
plt.xticks(resultDF['tenure'])
plt.grid()
plt.show()
```

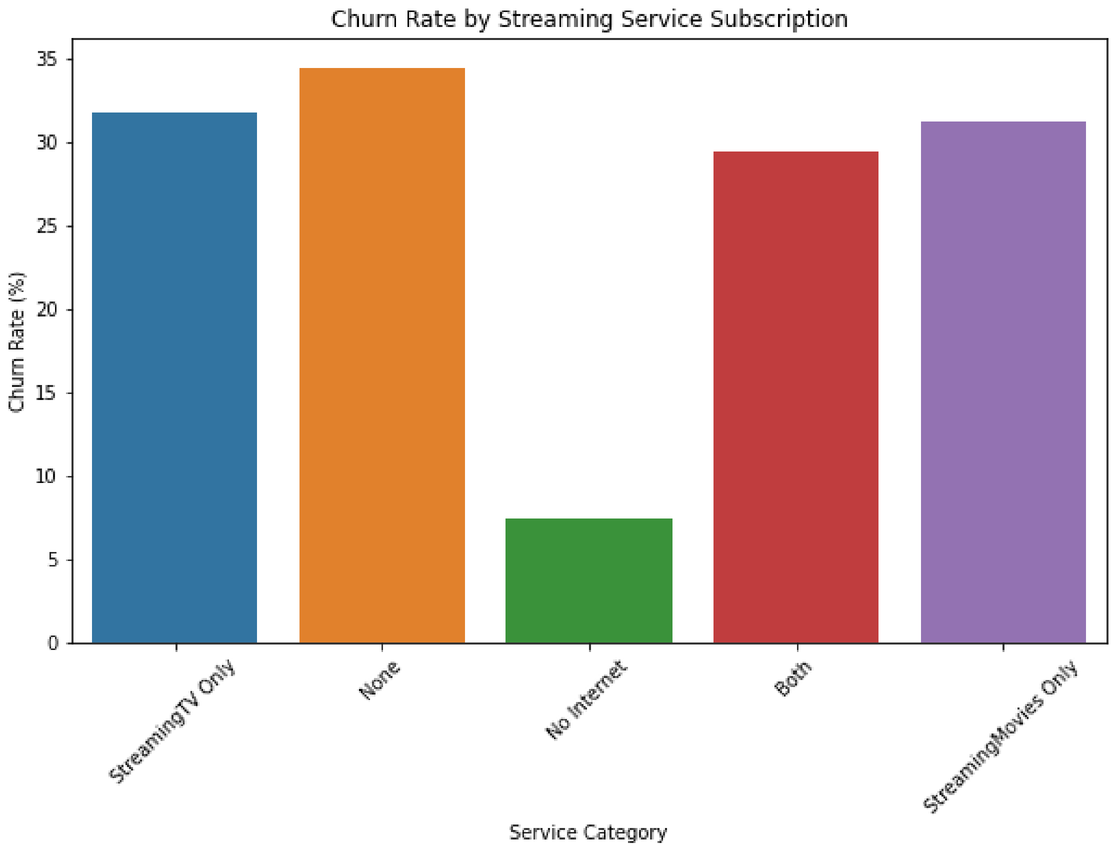


0.0.2 b) Investigate the churn rate of customers who subscribe to streaming services like StreamingTV and StreamingMovies.

[12]:
```
queryResult = spark.sql("""
    SELECT
        CASE
            WHEN StreamingTV = 'Yes' AND StreamingMovies = 'Yes' THEN 'Both'
            WHEN StreamingTV = 'Yes' AND StreamingMovies = 'No' THEN␣
↪'StreamingTV Only'
            WHEN StreamingTV = 'No' AND StreamingMovies = 'Yes' THEN␣
↪'StreamingMovies Only'
            WHEN StreamingTV = 'No' AND StreamingMovies = 'No' THEN 'None'
            ELSE 'No Internet'
        END AS service_category,
        COUNT(*) AS total_customers,
        SUM(CASE WHEN Churn = 'Yes' THEN 1 ELSE 0 END) AS churned_customers,
        ROUND(SUM(CASE WHEN Churn = 'Yes' THEN 1 ELSE 0 END) * 100.0 /␣
↪COUNT(*), 3) AS churn_rate
    FROM churnData
    GROUP BY service_category
""")
queryResult.show()
```

```
+--------------------+---------------+-----------------+----------+
|    service_category|total_customers|churned_customers|churn_rate|
+--------------------+---------------+-----------------+----------+
|    StreamingTV Only|            764|              243|    31.806|
|                None|           2017|              695|    34.457|
|         No Internet|           1520|              113|     7.434|
|                Both|           1939|              571|    29.448|
|StreamingMovies Only|            792|              247|    31.187|
+--------------------+---------------+-----------------+----------+
```

[13]:
```python
queryDF = queryResult.toPandas()
plt.figure(figsize=(10, 6))
sns.barplot(data=queryDF, x='service_category', y='churn_rate')
plt.title('Churn Rate by Streaming Service Subscription')
plt.xlabel('Service Category')
plt.ylabel('Churn Rate (%)')
plt.xticks(rotation=45)
plt.show()
```



0.0.3 c) Write Spark SQL to group customers by their tenure (e.g., 0-12 months, 13-24 months, etc.) and analyze churn rates in these tenure groups.

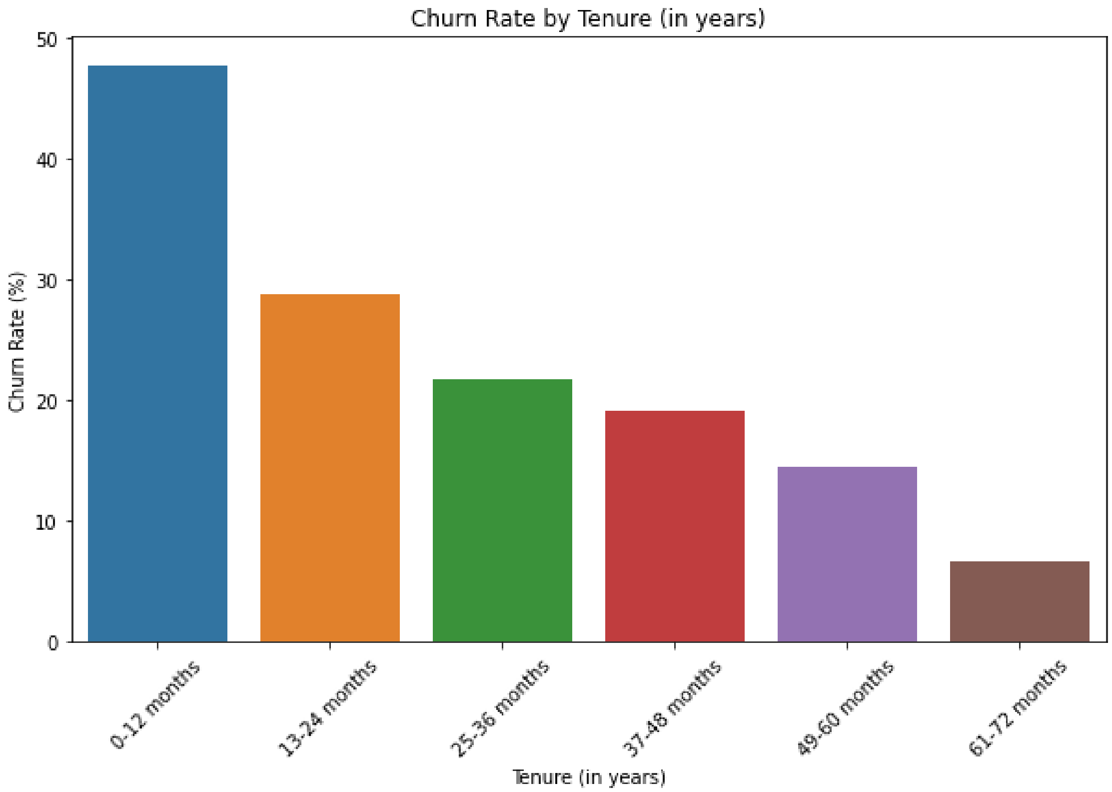
[14]:
```python
queryResult = spark.sql("""
    SELECT CASE
                WHEN tenure <= 12 THEN '0-12 months'
                WHEN tenure <= 24 THEN '13-24 months'
                WHEN tenure <= 36 THEN '25-36 months'
                WHEN tenure <= 48 THEN '37-48 months'
                WHEN tenure <= 60 THEN '49-60 months'
```

```
                WHEN tenure <= 72 THEN '61-72 months'
                ELSE '>72 months'
            END AS year,
            round(SUM(CASE WHEN Churn="Yes" THEN 1 ELSE 0 END)*100/COUNT(*) ,3)␣
 ↪AS churn_rate,
            COUNT(*) AS total
    FROM churnData
    GROUP BY year
    ORDER BY year
""")

queryResult.show()

# INSIGHT: churn_rate and tenure has an inverse relation
```

```
+------------+----------+-----+
|        year|churn_rate|total|
+------------+----------+-----+
| 0-12 months|    47.678| 2175|
|13-24 months|    28.711| 1024|
|25-36 months|    21.635|  832|
|37-48 months|    19.029|  762|
|49-60 months|    14.423|  832|
|61-72 months|      6.61| 1407|
+------------+----------+-----+
```

[15]:

```
queryDF = queryResult.toPandas()
plt.figure(figsize=(10, 6))
sns.barplot(data=queryDF, x='year', y='churn_rate')
plt.title('Churn Rate by Tenure (in years)')
plt.xlabel('Tenure (in years)')
plt.ylabel('Churn Rate (%)')
plt.xticks(rotation=45)
plt.show()
```



0.0.4 d) Analyze the impact of contract types and payment methods on churn rates.

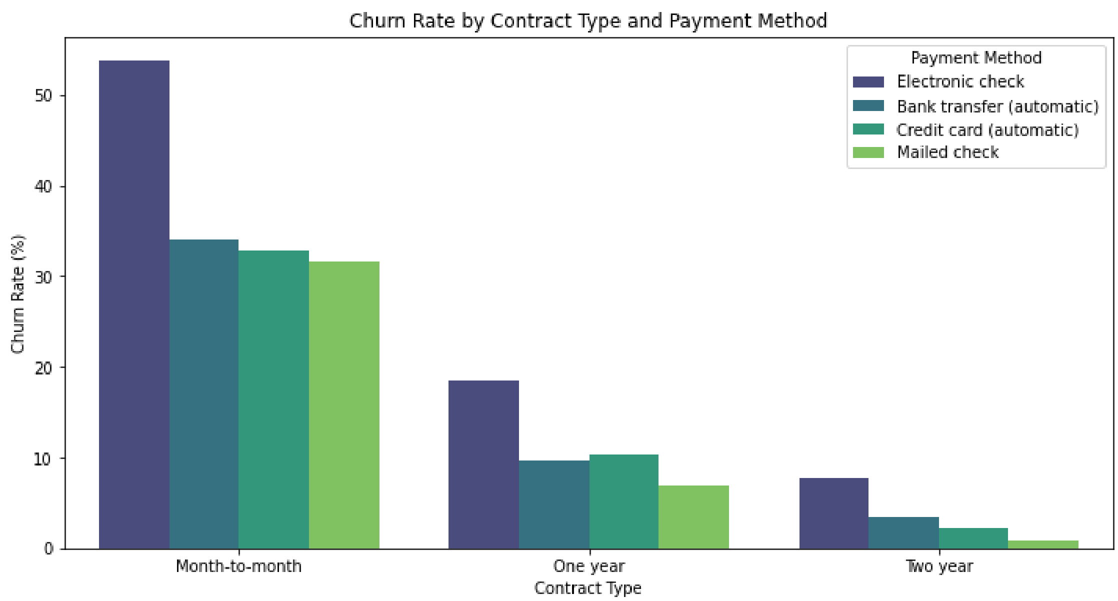
[16]:
```
queryResult = spark.sql("""
    SELECT Contract, PaymentMethod, round(SUM(CASE WHEN Churn="Yes" THEN 1 ELSE␣
↪0 END)*100/COUNT(*) ,3) AS churn_rate
    FROM churnData
    GROUP BY Contract, PaymentMethod
    ORDER BY churn_rate DESC
""")
queryResult.show()


# INSIGHT:
# 1. highest churn rate is for customers with shorter contract period␣
↪(month-to-month)
#    and customers with longer contract period has a generally lower churn rate
# 2. in each individual contract period type, customers using 'Electronic␣
↪Check' have generally
#    higher churn rate, and those using 'Mailed Check' have lower churn rate
```

```
+--------------+--------------------+----------+
```

```
|      Contract|       PaymentMethod|churn_rate|
+--------------+--------------------+----------+
|Month-to-month|    Electronic check|     53.73|
|Month-to-month|Bank transfer (au…|    34.126|
|Month-to-month|Credit card (auto…|    32.781|
|Month-to-month|        Mailed check|    31.579|
|      One year|    Electronic check|    18.444|
|      One year|Credit card (auto…|    10.302|
|      One year|Bank transfer (au…|     9.719|
|      Two year|    Electronic check|     7.738|
|      One year|        Mailed check|     6.845|
|      Two year|Bank transfer (au…|     3.381|
|      Two year|Credit card (auto…|     2.241|
|      Two year|        Mailed check|       0.8|
+--------------+--------------------+----------+
```

[17]:

```python
queryDF = queryResult.toPandas()
plt.figure(figsize=(12, 6))
sns.barplot(data=queryDF, x='Contract', y='churn_rate', hue='PaymentMethod',␣
↪palette='viridis')
plt.title('Churn Rate by Contract Type and Payment Method')
plt.xlabel('Contract Type')
plt.ylabel('Churn Rate (%)')
plt.legend(title='Payment Method')
plt.show()
```

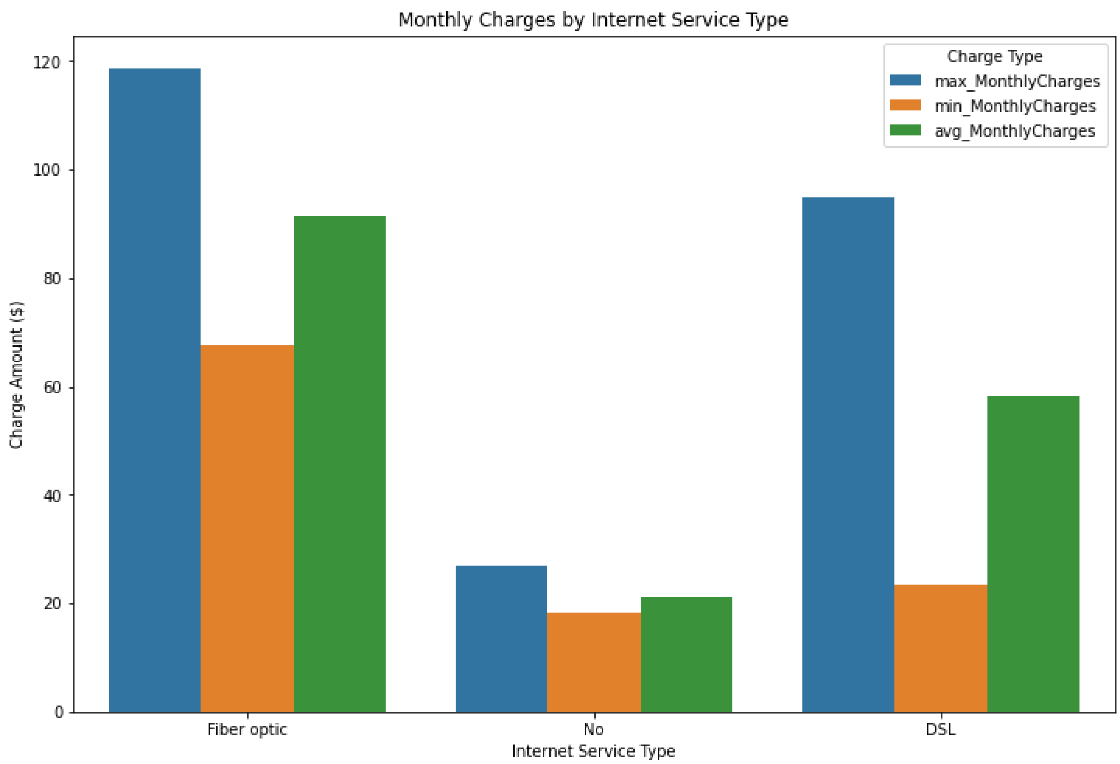


0.0.5 e) Explore the distribution of monthly charges for customers based on their type of internet service.

[18]:
```
queryResult = spark.sql("""
    SELECT InternetService,
           MAX(MonthlyCharges) AS max_MonthlyCharges,
           MIN(MonthlyCharges) AS min_MonthlyCharges,
           round(AVG(MonthlyCharges), 2) AS avg_MonthlyCharges,
           COUNT(*) AS total_customers
    FROM churnData
    GROUP BY InternetService
""")
queryResult.show()
```

```
+---------------+------------------+------------------+------------------+------
---------+
|InternetService|max_MonthlyCharges|min_MonthlyCharges|avg_MonthlyCharges|total_
customers|
+---------------+------------------+------------------+------------------+------
---------+
|    Fiber optic|            118.75|             67.75|              91.5|
3096|
|             No|              26.9|             18.25|             21.08|
1520|
|            DSL|              94.8|             23.45|             58.09|
2416|
+---------------+------------------+------------------+------------------+------
---------+
```

[19]:
```
queryDF = queryResult.toPandas()
queryDF = queryDF.melt(id_vars='InternetService',
                       value_vars=['max_MonthlyCharges', 'min_MonthlyCharges',␣
 ↪'avg_MonthlyCharges'],
                       var_name='Charge Type', value_name='Amount')

plt.figure(figsize=(12, 8))
sns.barplot(data=queryDF, x='InternetService', y='Amount', hue='Charge Type')
plt.title('Monthly Charges by Internet Service Type')
plt.xlabel('Internet Service Type')
plt.ylabel('Charge Amount ($)')
plt.legend(title='Charge Type')
plt.show()
```



0.0.6 f) Identify the top 10 customers who have contributed the most revenue to the company, based on total charges.

[20]:

```
queryResult = spark.sql("""
    SELECT customerID, TotalCharges
    FROM churnData
    ORDER BY TotalCharges DESC
""")
queryResult.limit(10).show()
```

```
+----------+------------+
|customerID|TotalCharges|
+----------+------------+
|2889-FPWRM|      8684.8|
|7569-NMZYQ|     8672.45|
|9739-JLPQJ|      8670.1|
|9788-HNGUT|      8594.4|
|8879-XUAHX|     8564.75|
|9924-JPRMC|     8547.15|
|0675-NCDYU|     8543.25|
|6650-BWFRT|      8529.5|
|0164-APGRB|      8496.7|
```

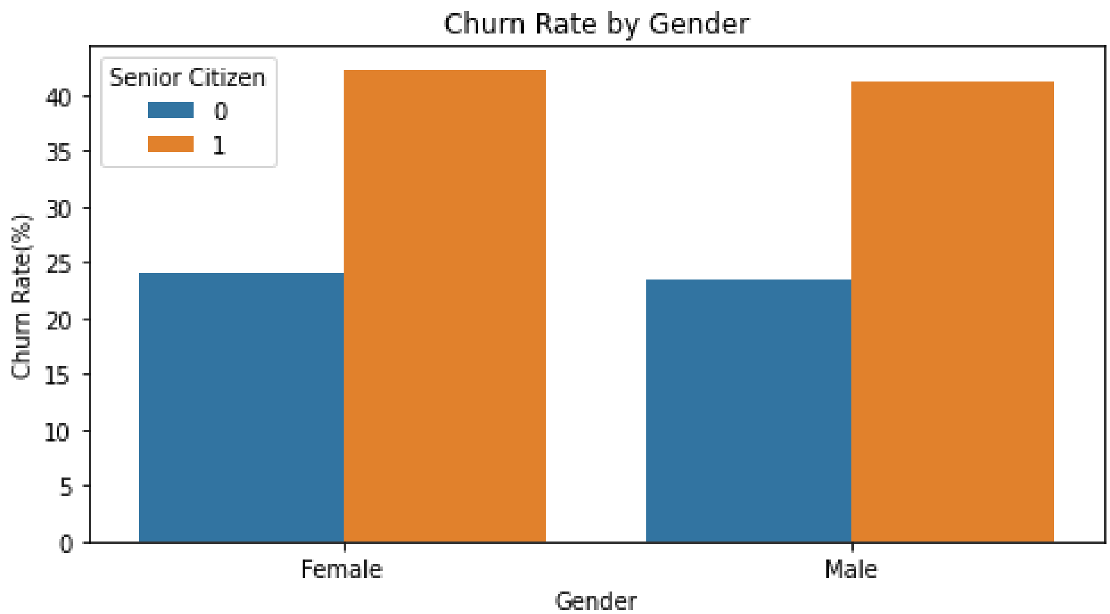
```
|1488-PBLJN|      8477.7|
+----------+------------+
```

0.0.7 g) Calculate the churn rate segmented by gender and whether the customer is a senior citizen.

[21]:
```
queryResult = spark.sql("""
    SELECT gender, SeniorCitizen,
           round(SUM(CASE WHEN Churn="Yes" THEN 1 ELSE 0 END)*100/COUNT(*) ,3)␣
 ↪AS churn_rate
    FROM churnData
    GROUP BY gender, SeniorCitizen
    ORDER BY churn_rate DESC
""")
queryResult.show()

# INSIGHT: senior citizens have higher churn rate
```

```
+------+-------------+----------+
|gender|SeniorCitizen|churn_rate|
+------+-------------+----------+
|Female|            1|    42.254|
|  Male|            1|    41.115|
|Female|            0|    23.979|
|  Male|            0|    23.328|
+------+-------------+----------+
```

[22]:
```
queryDF = queryResult.toPandas()

plt.figure(figsize=(8, 4))
sns.barplot(data=queryDF, x='gender', y='churn_rate', hue='SeniorCitizen')
plt.title('Churn Rate by Gender')
plt.xlabel('Gender')
plt.ylabel('Churn Rate(%)')
plt.legend(title='Senior Citizen')
plt.show()
```

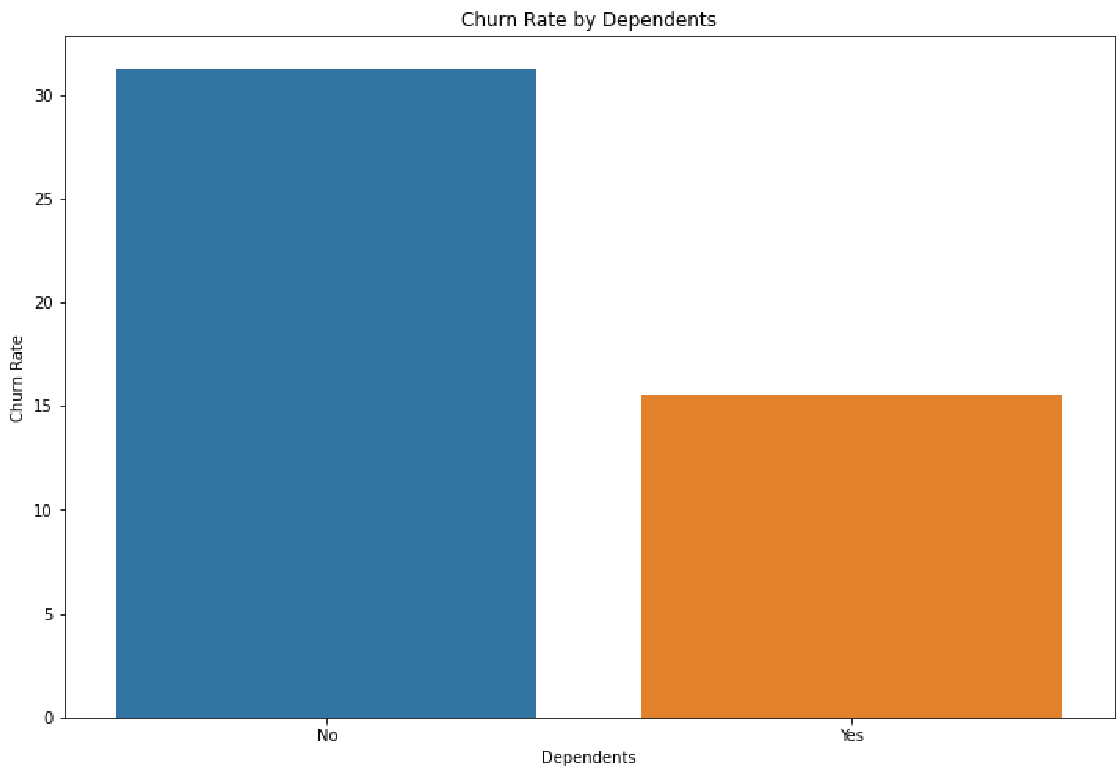


0.0.8 h) Write query to calculate Correlation between dependents and churn. Explore whether having dependents affects customer churn rates.

[23]:
```
queryResult = spark.sql("""
    SELECT Dependents,
           round(SUM(CASE WHEN Churn="Yes" THEN 1 ELSE 0 END)*100/COUNT(*) ,3)␣
 ↪AS churn_rate
    FROM churnData
    GROUP BY Dependents
    ORDER BY churn_rate DESC
""")
queryResult.show()

# INSIGHT: customers without dependents have a much higher churn rate than␣
 ↪those with dependents
```

```
+----------+----------+
|Dependents|churn_rate|
+----------+----------+
|        No|    31.279|
|       Yes|    15.531|
+----------+----------+
```

[24]:
```
queryDF = queryResult.toPandas()

plt.figure(figsize=(12, 8))
sns.barplot(data=queryDF, x='Dependents', y='churn_rate')
plt.title('Churn Rate by Dependents')
plt.xlabel('Dependents')
plt.ylabel('Churn Rate')
plt.show()
```



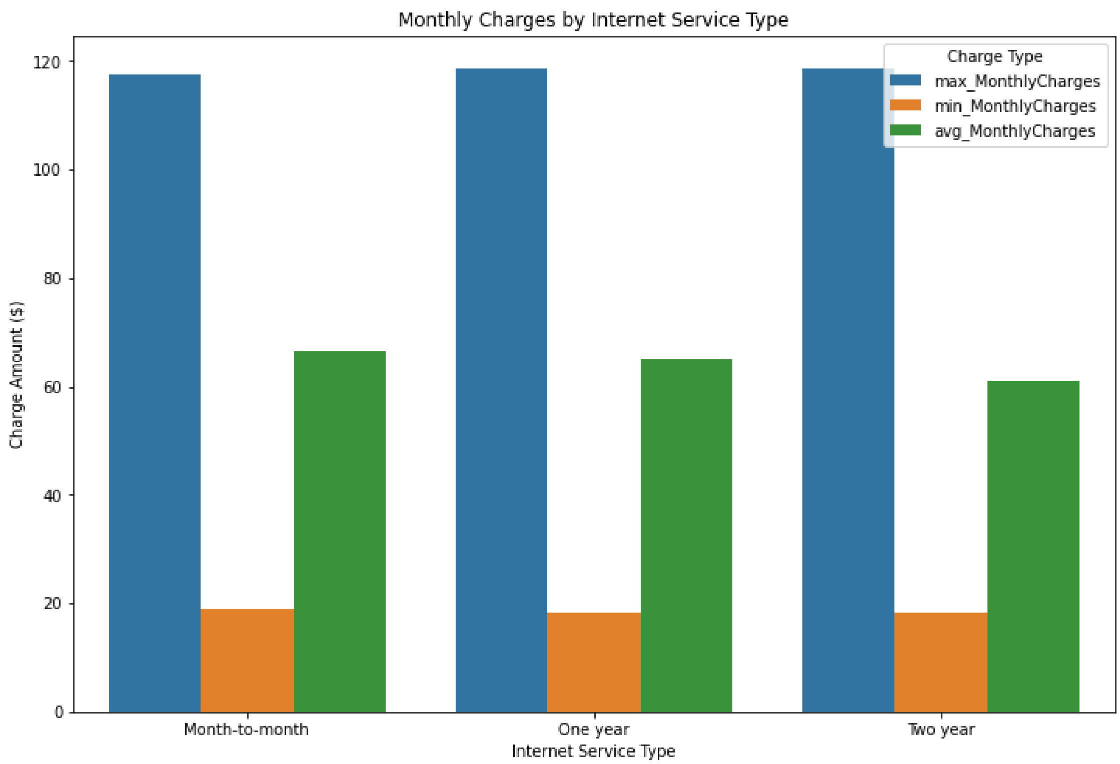
0.0.9 i) Predict potential churn rates by analyzing the relationship between monthly charges, contract types, and the churn rate.

[94]:
```
queryResult = spark.sql("""
    SELECT Contract,
           MIN(MonthlyCharges) AS min_MonthlyCharges,
           MAX(MonthlyCharges) AS max_MonthlyCharges,
           round(AVG(MonthlyCharges), 3) AS avg_MonthlyCharges,
           round(SUM(CASE WHEN Churn="Yes" THEN 1 ELSE 0 END)*100/COUNT(*) ,3)␣
↪AS churn_rate
    FROM churnData
    GROUP BY Contract
```

```
    ORDER BY churn_rate DESC
""")
queryResult.show()

# churn_rate is directly related to avg_MonthlyCharges
# as contract period increases, average monthly charges decreases and churn␣
↪rate also decreases
```

```
+--------------+-----------------+-----------------+-----------------+-------
---+
|
Contract|min_MonthlyCharges|max_MonthlyCharges|avg_MonthlyCharges|churn_rate|
+--------------+-----------------+-----------------+-----------------+-------
---+
|Month-to-month|             18.75|            117.45|            66.398|
42.71|
|      One year|             18.25|             118.6|            65.079|
11.277|
|      Two year|              18.4|            118.75|            60.872|
2.849|
+--------------+-----------------+-----------------+-----------------+-------
---+
```

[95]:
```
queryDF = queryResult.toPandas()
queryDF = queryDF.melt(id_vars='Contract', value_vars=['max_MonthlyCharges',␣
↪'min_MonthlyCharges', 'avg_MonthlyCharges'],
                            var_name='Contract Type', value_name='Amount')

plt.figure(figsize=(12, 8))
sns.barplot(data=queryDF, x='Contract', y='Amount', hue='Contract Type')
plt.title('Monthly Charges by Internet Service Type')
plt.xlabel('Internet Service Type')
plt.ylabel('Charge Amount ($)')
plt.legend(title='Charge Type')
plt.show()
```



0.0.10 j) Determine the churn rate for customers who have multiple services (Phone, Internet, and Streaming), which can help understand whether bundling services leads to higher or lower churn. Calculate churn rate for customers with multiple services.

[27]:

```
queryResult = spark.sql("""
    SELECT PhoneService, InternetService, StreamingTV, StreamingMovies,
           round(SUM(CASE WHEN Churn="Yes" THEN 1 ELSE 0 END)*100/COUNT(*) ,3)␣
 ↪AS churn_rate
    FROM churnData
    GROUP BY PhoneService, InternetService, StreamingTV, StreamingMovies
    ORDER BY churn_rate DESC
""")
queryResult.show()


# INSIGHT:
# 1. customers with Phone service and fiber optic internet service have high␣
 ↪churn rates
#    and presence of Streaming services reduces churn rate
```

```
# 2. While customers with Phone service and DSL internet service have lower␣
↪churn rates
#    and presence of Streaming services reduces churn rate
# 3. Customers with no phone service have all opted for DSL internet. and␣
↪presence of
#    streaming services decreases the churn rate
```

```
+------------+---------------+-------------------+-------------------+----------
+
|PhoneService|InternetService|        StreamingTV|
StreamingMovies|churn_rate|
+------------+---------------+-------------------+-------------------+----------
+
|         Yes|    Fiber optic|                 No|                 No|
46.63|
|         Yes|    Fiber optic|                Yes|                 No|
44.091|
|         Yes|    Fiber optic|                 No|                Yes|
42.63|
|         Yes|    Fiber optic|                Yes|                Yes|
37.634|
|          No|            DSL|                 No|                Yes|
29.592|
|          No|            DSL|                 No|                 No|
25.753|
|          No|            DSL|                Yes|                 No|
25.301|
|         Yes|            DSL|                 No|                 No|
24.108|
|          No|            DSL|                Yes|                Yes|
21.5|
|         Yes|            DSL|                 No|                Yes|
11.858|
|         Yes|            DSL|                Yes|                 No|
11.618|
|         Yes|            DSL|                Yes|                Yes|
8.159|
|         Yes|             No|No internet service|No internet service|
7.434|
+------------+---------------+-------------------+-------------------+----------
+
```

```
[ ]:
```

0.0.11 k) Churn Impact by device protection and online backup services. Write query to investigate whether having device protection or online backup services affects churn rates.

[28]:
```
spark.sql("""
    SELECT DeviceProtection, OnlineBackup, OnlineSecurity,
           round(SUM(CASE WHEN Churn="Yes" THEN 1 ELSE 0 END)*100/COUNT(*) ,3)␣
 ↪AS churn_rate
    FROM churnData
    GROUP BY DeviceProtection, OnlineBackup, OnlineSecurity
    ORDER BY churn_rate DESC
""").show()

# INSIGHT: Customers who have opted for none of the services show highest churn␣
 ↪rates,
# while those with all of the above mentioned services have lower churn rates
# owest churn rate is for customers who have no internet services
```

```
+-------------------+-------------------+-------------------+----------+
|   DeviceProtection|       OnlineBackup|     OnlineSecurity|churn_rate|
+-------------------+-------------------+-------------------+----------+
|                 No|                 No|                 No|    52.551|
|                Yes|                 No|                 No|    38.484|
|                 No|                Yes|                 No|    34.808|
|                Yes|                Yes|                 No|    26.923|
|                 No|                 No|                Yes|    24.842|
|                 No|                Yes|                Yes|    14.815|
|                Yes|                 No|                Yes|    13.909|
|                Yes|                Yes|                Yes|     7.959|
|No internet service|No internet service|No internet service|     7.434|
+-------------------+-------------------+-------------------+----------+
```

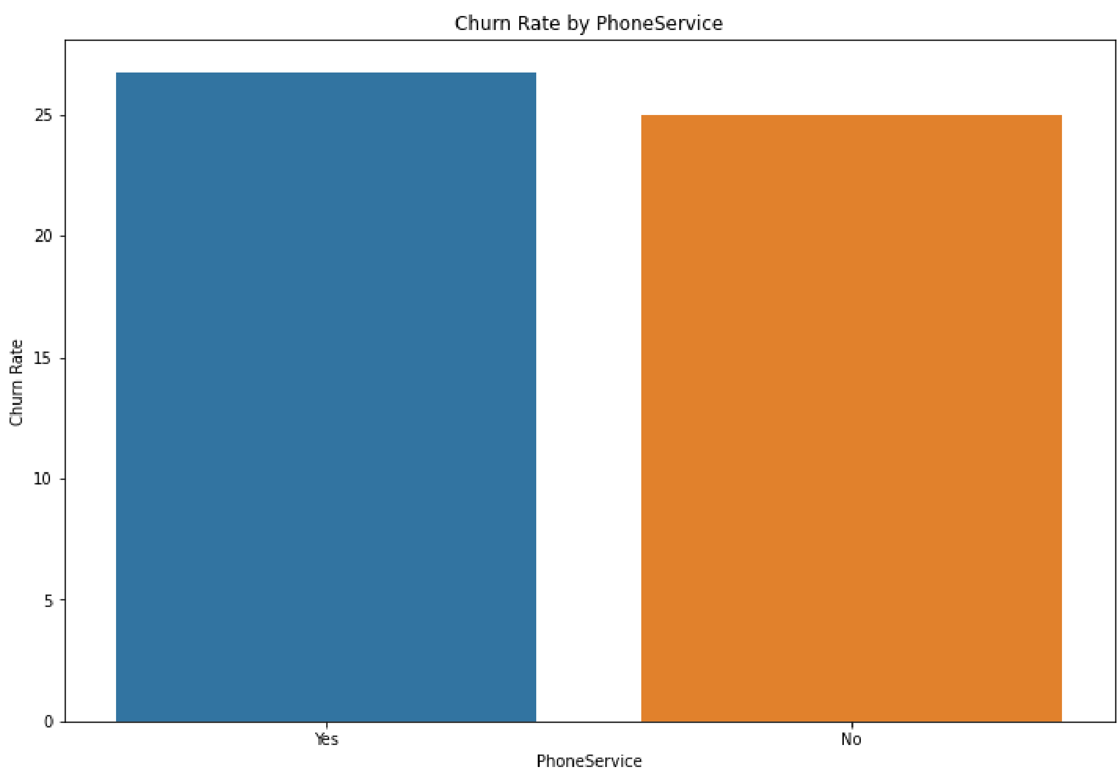
0.0.12 l) Explore churn rates among customers who do not have phone service and investigate if it influences customer retention.

[29]:
```
queryResult = spark.sql("""
    SELECT PhoneService,
           round(SUM(CASE WHEN Churn="Yes" THEN 1 ELSE 0 END)*100/COUNT(*) ,3)␣
 ↪AS churn_rate,
           COUNT(*) AS total
    FROM churnData
    GROUP BY PhoneService
    ORDER BY churn_rate DESC
""")
queryResult.show()
```

```
# INSIGHT: churn rates are higher by a small margin for customers with␣
 ↪PhoneService
# but it is to be noted that there are 10 times more customers who have opted␣
 ↪in for PhoneService
```

```
+------------+----------+-----+
|PhoneService|churn_rate|total|
+------------+----------+-----+
|         Yes|    26.747| 6352|
|          No|      25.0|  680|
+------------+----------+-----+
```

[30]:
```
queryDF = queryResult.toPandas()

plt.figure(figsize=(12, 8))
sns.barplot(data=queryDF, x='PhoneService', y='churn_rate')
plt.title('Churn Rate by PhoneService')
plt.xlabel('PhoneService')
plt.ylabel('Churn Rate')
plt.show()
```

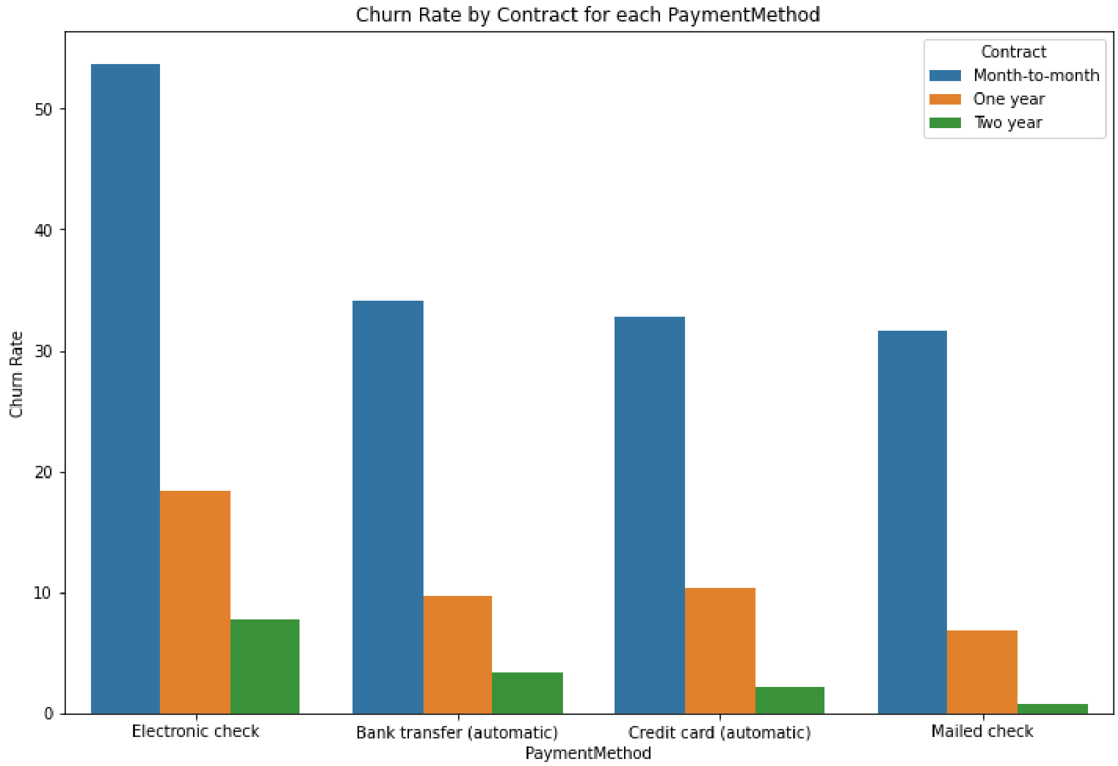


0.0.13 m) Understand the relationship between payment methods and contract types on customer churn. This query will help you discover which combinations are most prone to churn.

[31]:
```
queryResult = spark.sql("""
    SELECT PaymentMethod, Contract,
           round(SUM(CASE WHEN Churn="Yes" THEN 1 ELSE 0 END)*100/COUNT(*) ,3)␣
 ↪AS churn_rate
    FROM churnData
    GROUP BY PaymentMethod, Contract
    ORDER BY churn_rate DESC
""")
queryResult.show()

# INSIGHT: customers with a shorter contract period has higher churn rate␣
 ↪compared to those with longer
# contract periods
```

```
+--------------------+--------------+----------+
|       PaymentMethod|      Contract|churn_rate|
+--------------------+--------------+----------+
|    Electronic check|Month-to-month|     53.73|
|Bank transfer (au…|Month-to-month|    34.126|
|Credit card (auto…|Month-to-month|    32.781|
|        Mailed check|Month-to-month|    31.579|
|    Electronic check|      One year|    18.444|
|Credit card (auto…|      One year|    10.302|
|Bank transfer (au…|      One year|     9.719|
|    Electronic check|      Two year|     7.738|
|        Mailed check|      One year|     6.845|
|Bank transfer (au…|      Two year|     3.381|
|Credit card (auto…|      Two year|     2.241|
|        Mailed check|      Two year|       0.8|
+--------------------+--------------+----------+
```

[32]:
```
queryDF = queryResult.toPandas()

plt.figure(figsize=(12, 8))
sns.barplot(data=queryDF, x='PaymentMethod', y='churn_rate', hue='Contract')
plt.title('Churn Rate by Contract for each PaymentMethod')
plt.xlabel('PaymentMethod')
plt.ylabel('Churn Rate')
plt.show()
```



0.0.14 n) Analyze how customer churn is affected by senior citizen status and whether the customer has dependents.

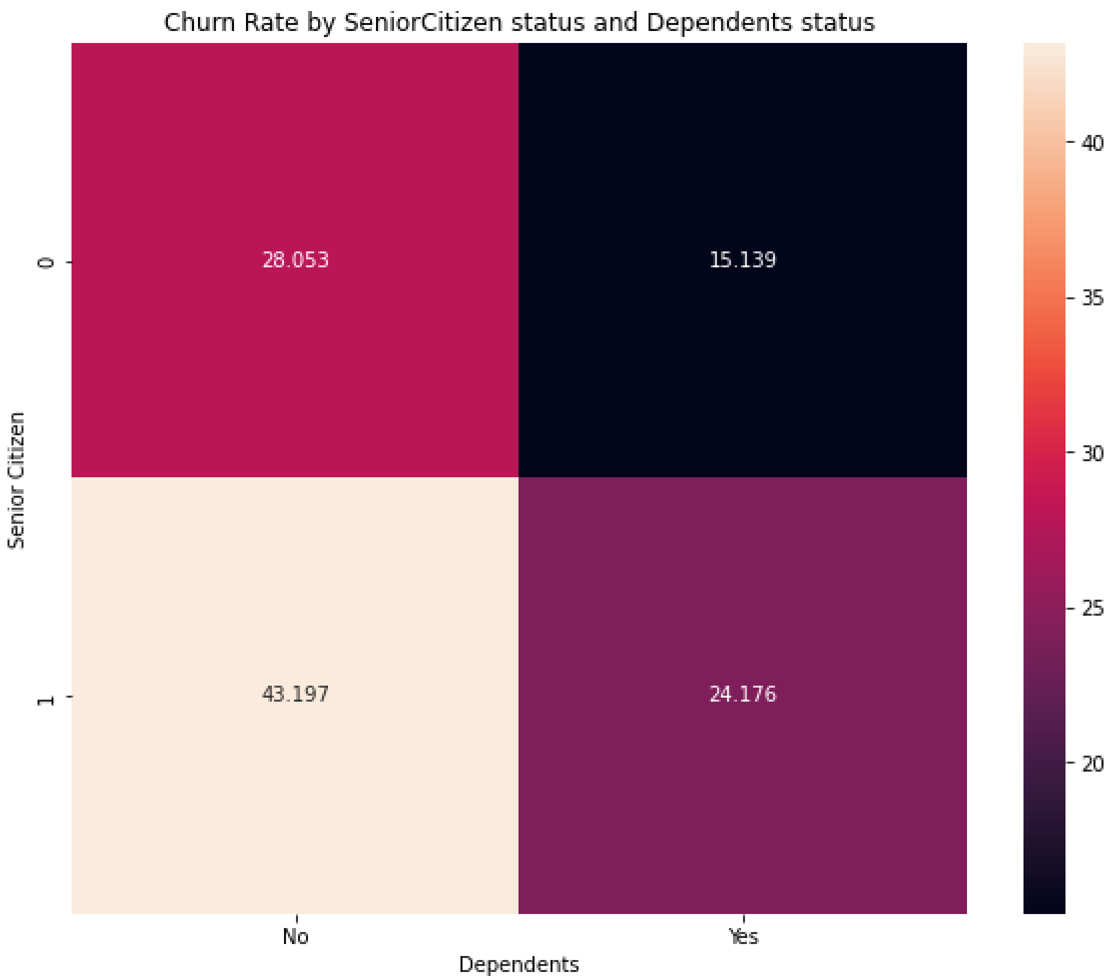
[96]:
```
queryResult = spark.sql("""
    SELECT SeniorCitizen, Dependents,
           round(SUM(CASE WHEN Churn="Yes" THEN 1 ELSE 0 END)*100/COUNT(*) ,3)␣
 ↪AS churn_rate
    FROM churnData
    GROUP BY SeniorCitizen, Dependents
    ORDER BY churn_rate DESC
""")
queryResult.show()

# highest churn rate is for senior citizens with no dependents
# while lowest churn rate is for non-senior ciitzens with dependents
```

```
+-------------+----------+----------+
|SeniorCitizen|Dependents|churn_rate|
+-------------+----------+----------+
|            1|        No|    43.197|
|            0|        No|    28.053|
```

```
|            1|       Yes|    24.176|
|            0|       Yes|    15.139|
+-------------+----------+----------+
```

[98]:

```
queryDF = queryResult.toPandas()

plt.figure(figsize=(10, 8))
sns.heatmap(data=queryDF.pivot("SeniorCitizen", "Dependents", "churn_rate"),␣
 ↪annot=True, fmt='.3f')
plt.title('Churn Rate by SeniorCitizen status and Dependents status')
plt.xlabel('Dependents')
plt.ylabel('Senior Citizen')
plt.show()
```



0.0.15 o) Explore whether subscribing to streaming services like Streaming TV and Streaming Movies influences the churn rate.

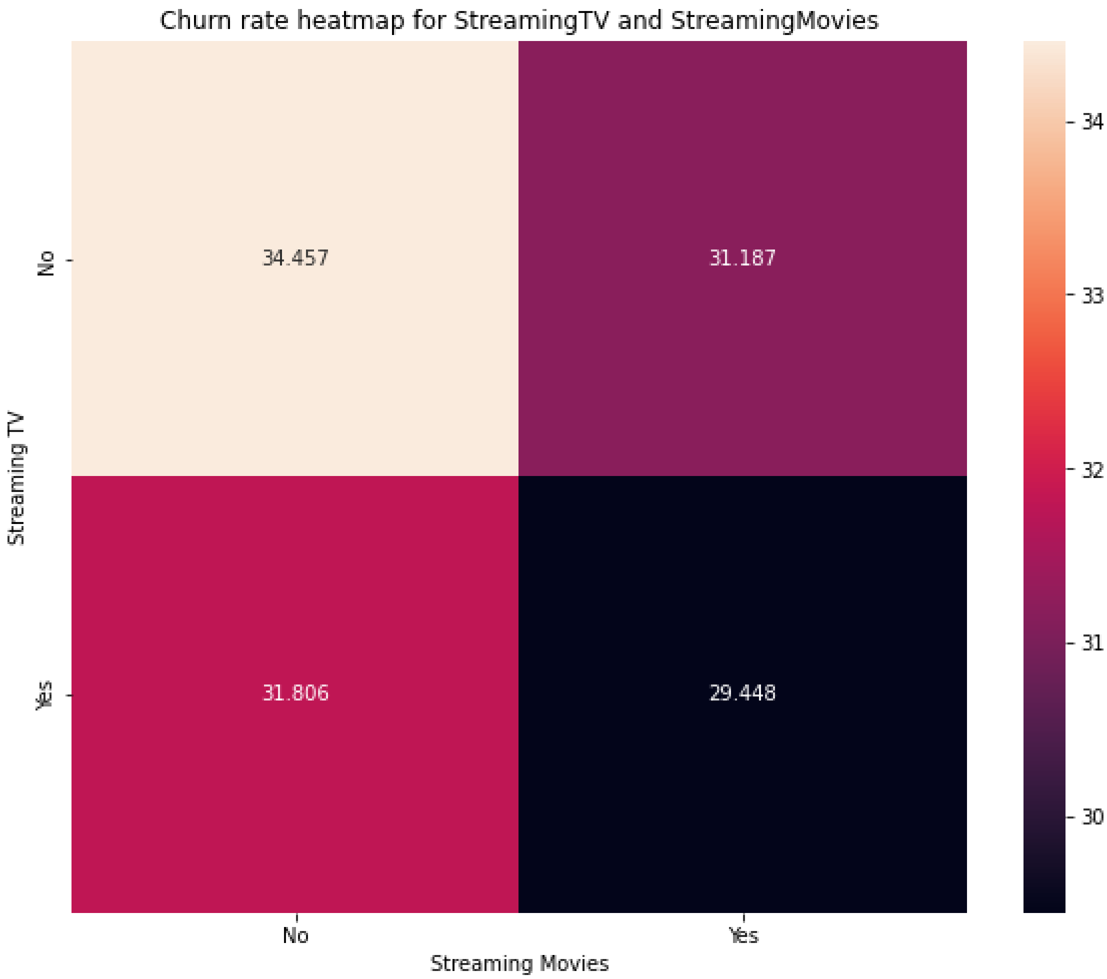
[100]:

```
queryResult = spark.sql("""
    SELECT StreamingTV, StreamingMovies,
           round(SUM(CASE WHEN Churn="Yes" THEN 1 ELSE 0 END)*100/COUNT(*), 3)␣
 ↪AS churn_rate,
           COUNT(*) AS count
    FROM churnData
    WHERE StreamingTV="Yes" OR StreamingMovies="Yes" OR StreamingTV="No" OR␣
 ↪StreamingMovies="No"
    GROUP BY StreamingTV, StreamingMovies
""")
queryResult.show()

# INSIGHT: churn rate is higher for customers who haven't chosen␣
 ↪StreamingMovies and StreamingTV
```

```
+-----------+---------------+----------+-----+
|StreamingTV|StreamingMovies|churn_rate|count|
+-----------+---------------+----------+-----+
|        Yes|            Yes|    29.448| 1939|
|         No|             No|    34.457| 2017|
|        Yes|             No|    31.806|  764|
|         No|            Yes|    31.187|  792|
+-----------+---------------+----------+-----+
```

[101]:

```
queryDF = queryResult.toPandas()

plt.figure(figsize=(10, 8))
sns.heatmap(data=queryDF.pivot("StreamingTV", "StreamingMovies", "churn_rate"),␣
 ↪annot=True, fmt='.3f')
plt.title("Churn rate heatmap for StreamingTV and StreamingMovies")
plt.xlabel("Streaming Movies")
plt.ylabel("Streaming TV")
plt.show()
```



0.0.16 p) Understand how tenure and MonthlyCharges differ between churned and non-churned customers. This can provide insights into the behavior of long-term customers.

[102]:
```
spark.sql("""
    SELECT Churn,
           round(AVG(MonthlyCharges), 3) AS avg_MonthlyCharges,
           round(AVG(Tenure), 3) AS avg_Tenure
    FROM churnData
    GROUP BY Churn
""").show()
```

```
+-----+------------------+----------+
|Churn|avg_MonthlyCharges|avg_Tenure|
+-----+------------------+----------+
|   No|            61.307|     37.65|
```

```
|  Yes|            74.441|    17.979|
+-----+------------------+----------+
```

0.0.17 q) Compare monthly charges and churn rates between newer customers and long-time customers.

[38]:

```
# assuming that customers with tenure<=12 as newer customers
# and those with a tenure >36 months as long-time customers

# spark.sql("""
#     SELECT
#         CASE
#             WHEN median_tenure_value = 1 THEN 'Newer'
#             WHEN median_tenure_value = 2 THEN 'Long-time'
#         END AS customer_category,
#         round(AVG(MonthlyCharges), 3) AS avg_MonthlyCharges,
#         round(SUM(CASE WHEN Churn="Yes" THEN 1 ELSE 0 END)*100/COUNT(*), 3)␣
↪AS churn_rate,
#         COUNT(*) AS total
#     FROM (
#         SELECT tenure, MonthlyCharges, Churn,
#         NTILE(2) OVER(ORDER BY tenure) as median_tenure_value
#         FROM churnData
#     ) AS _
#     GROUP BY customer_category
# """).show()
spark.sql("""
    SELECT
        CASE
            WHEN Tenure <= 12 THEN 'Newer'
            WHEN Tenure > 36 THEN 'Long-time'
            ELSE 'Others'
        END AS customer_category,
        ROUND(AVG(MonthlyCharges), 3) AS avg_MonthlyCharges,
        ROUND(SUM(CASE WHEN Churn = 'Yes' THEN 1 ELSE 0 END) * 100 / COUNT(*),␣
↪3) AS churn_rate,
        COUNT(*) AS total
    FROM churnData
    GROUP BY customer_category
""").show()



# INSIGHT:
```

```
# customers classified as newer customer have a high churn rate and lower␣
↪average monthly charges
# while customer classified as long-time have lower churn rate and higher␣
↪average monthly charges
# therefore, average monthly charges are directly related to tenure
# while churn rate is inversely related
```

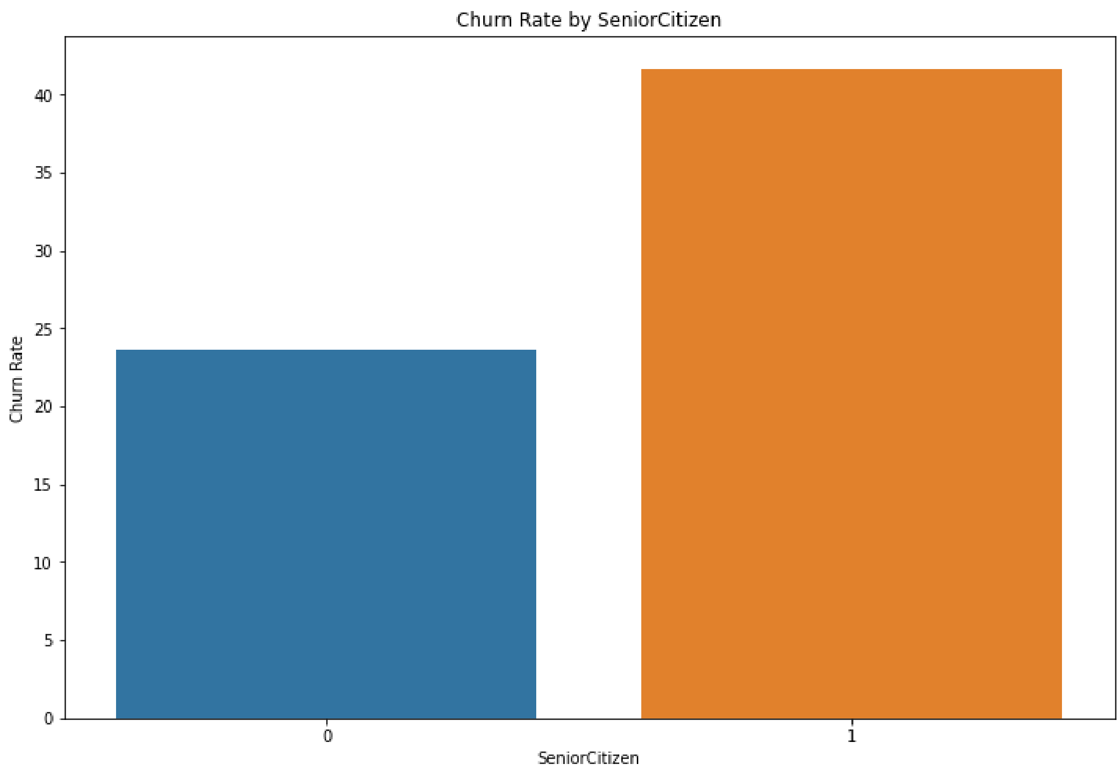
```
+-----------------+-----------------+----------+-----+
|customer_category|avg_MonthlyCharges|churn_rate|total|
+-----------------+-----------------+----------+-----+
|        Long-time|           72.009|    11.929| 3001|
|            Newer|           56.172|    47.678| 2175|
|           Others|           63.248|    25.539| 1856|
+-----------------+-----------------+----------+-----+
```

0.0.18 r) What is the correlation between senior citizen status and churn rate?

[39]:
```
queryResult = spark.sql("""
    SELECT SeniorCitizen,
           round(SUM(CASE WHEN Churn="Yes" THEN 1 ELSE 0 END)*100/COUNT(*) ,3)␣
↪AS churn_rate
    FROM churnData
    GROUP BY SeniorCitizen
    ORDER BY churn_rate DESC
""")
queryResult.show()

# INSIGHT: Senior citizen have a much higher churn rate than others
```

```
+-------------+----------+
|SeniorCitizen|churn_rate|
+-------------+----------+
|            1|    41.681|
|            0|     23.65|
+-------------+----------+
```

[40]:
```
queryDF = queryResult.toPandas()
plt.figure(figsize=(12, 8))
sns.barplot(data=queryDF, x='SeniorCitizen', y='churn_rate')
plt.title('Churn Rate by SeniorCitizen')
plt.xlabel('SeniorCitizen')
plt.ylabel('Churn Rate')
plt.show()
```



0.0.19 s) Partition customers based on whether they are senior citizens and divide them into 5 groups based on tenure. [Use NTILE.]

[41]:
```
spark.sql("""
    SELECT customerID, SeniorCitizen, tenure,
           NTILE(5) OVER(PARTITION BY SeniorCitizen ORDER BY tenure) as ntile
    FROM churnData
""").show()

# INSIGHT: below table shows data partitioned based on SeniorCitizen status and
# divided into 5 groups based on tenure
```

```
+----------+-------------+------+-----+
|customerID|SeniorCitizen|tenure|ntile|
+----------+-------------+------+-----+
|8779-QRDMV|            1|     1|    1|
|3413-BMNZE|            1|     1|    1|
|2424-WVHPL|            1|     1|    1|
|0390-DCFDQ|            1|     1|    1|
|9514-JDSKI|            1|     1|    1|
|0021-IKXGC|            1|     1|    1|
```

```
|5564-NEMQO|            1|     1|    1|
|5192-EBGOV|            1|     1|    1|
|6513-EECDB|            1|     1|    1|
|7206-GZCDC|            1|     1|    1|
|1768-ZAIFU|            1|     1|    1|
|6567-HOOPW|            1|     1|    1|
|5240-IJOQT|            1|     1|    1|
|0661-XEYAN|            1|     1|    1|
|3068-OMWZA|            1|     1|    1|
|8580-AECUZ|            1|     1|    1|
|5047-LHVLY|            1|     1|    1|
|8375-DKEBR|            1|     1|    1|
|8080-DDEMJ|            1|     1|    1|
|6702-OHFWR|            1|     1|    1|
+----------+-------------+------+-----+
only showing top 20 rows
```

0.0.20 t) Use PERCENT_RANK to identify the top 5% of customers by MonthlyCharges.

[109]:
```
spark.sql("""
    SELECT customerID, MonthlyCharges, percent_rank
    FROM (
        SELECT customerID, MonthlyCharges,
               PERCENT_RANK() OVER(ORDER BY MonthlyCharges DESC) AS percent_rank
        FROM churnData
    ) AS _
    WHERE percent_rank <= 0.05
""").show(truncate=False)

# INSIGHT: below table shows top 5% of customers who pay the highest␣
↪MonthlyCharges
```

```
+----------+--------------+---------------------+
|customerID|MonthlyCharges|percent_rank         |
+----------+--------------+---------------------+
|7569-NMZYQ|118.75        |0.0                  |
|8984-HPEMB|118.65        |1.4222727919214906E-4|
|5989-AXPUC|118.6         |2.844545583842981E-4 |
|5734-EJKXG|118.6         |2.844545583842981E-4 |
|8199-ZLLSA|118.35        |5.689091167685962E-4 |
|9924-JPRMC|118.2         |7.111363959607452E-4 |
|2889-FPWRM|117.8         |8.533636751528943E-4 |
|3810-DVDQQ|117.6         |9.955909543450433E-4 |
|9739-JLPQJ|117.5         |0.0011378182335371925|
|2302-ANTDP|117.45        |0.0012800455127293415|
```

```
|6904-JLBGY|117.35        |0.0014222727919214904|
|4282-MSACW|117.2         |0.0015645000711136396|
|6650-BWFRT|117.15        |0.0017067273503057886|
|9788-HNGUT|116.95        |0.0018489546294979378|
|1488-PBLJN|116.85        |0.0019911819086900866|
|0017-IUDMW|116.8         |0.00213340918788223 6 |
|8628-MFKAX|116.75        |0.0022756364670743 85 |
|3680-CTHUH|116.6         |0.002417863746266534 |
|3258-ZKPAI|116.6         |0.002417863746266534 |
|3795-CAWEX|116.55        |0.002702318304650832 |
+----------+--------------+---------------------+
only showing top 20 rows
```

0.0.21 u) Find customers who fall within the top 5% of the distribution based on monthly charges. Compare total charges with the next customer in the same internet service type, based on monthly charges.

[111]:

```
spark.sql("""
    SELECT *, round(TotalCharges-next_TotalCharges, 3) AS difference
    FROM (
        SELECT customerID, InternetService, MonthlyCharges, TotalCharges,
        LEAD(TotalCharges, 1) OVER (PARTITION BY InternetService ORDER BY␣
 ↪TotalCharges DESC) AS next_TotalCharges
        FROM (
            SELECT customerID, MonthlyCharges, InternetService, TotalCharges,
                   PERCENT_RANK() OVER(ORDER BY MonthlyCharges DESC) AS␣
 ↪percent_rank
            FROM churnData
        ) AS _
        WHERE percent_rank <= 0.05
    ) AS _
""").show()



# INSIGHT: below table shows data partitioned based on InternetService type and␣
 ↪ordered in
# descending order of MonthlyCharges for top 5% of customers.
```

```
+----------+---------------+--------------+------------+-----------------+------
----+
|customerID|InternetService|MonthlyCharges|TotalCharges|next_TotalCharges|differ
ence|
+----------+---------------+--------------+------------+-----------------+------
----+
```

```
|2889-FPWRM|    Fiber optic|         117.8|      8684.8|           8672.45|
12.35|
|7569-NMZYQ|    Fiber optic|        118.75|     8672.45|            8670.1|
2.351|
|9739-JLPQJ|    Fiber optic|         117.5|      8670.1|            8594.4|
75.699|
|9788-HNGUT|    Fiber optic|        116.95|      8594.4|           8564.75|
29.65|
|8879-XUAHX|    Fiber optic|        116.25|     8564.75|           8547.15|
17.6|
|9924-JPRMC|    Fiber optic|         118.2|     8547.15|           8543.25|
3.9|
|0675-NCDYU|    Fiber optic|         116.4|     8543.25|            8529.5|
13.75|
|6650-BWFRT|    Fiber optic|        117.15|      8529.5|            8496.7|
32.8|
|0164-APGRB|    Fiber optic|         114.9|      8496.7|            8477.7|
19.0|
|1488-PBLJN|    Fiber optic|        116.85|      8477.7|            8477.6|
0.101|
|8984-HPEMB|    Fiber optic|        118.65|      8477.6|            8476.5|
1.1|
|6007-TCTST|    Fiber optic|         115.8|      8476.5|            8468.2|
8.3|
|4376-KFVRS|    Fiber optic|        114.05|      8468.2|           8456.75|
11.45|
|0017-IUDMW|    Fiber optic|         116.8|     8456.75|            8443.7|
13.05|
|5451-YHYPW|    Fiber optic|        115.75|      8443.7|           8436.25|
7.45|
|6904-JLBGY|    Fiber optic|        117.35|     8436.25|            8425.3|
10.95|
|8263-QMNTJ|    Fiber optic|        115.55|      8425.3|           8425.15|
0.149|
|8015-IHCGW|    Fiber optic|         115.5|     8425.15|            8424.9|
0.25|
|5914-XRFQB|    Fiber optic|         115.8|      8424.9|            8405.0|
19.9|
|8454-AATJP|    Fiber optic|        115.05|      8405.0|            8404.9|
0.1|
+----------+---------------+--------------+------------+------------------+------
----+
only showing top 20 rows
```

0.0.22 v) Find the top 5 customers with the highest MonthlyCharges within each Contract type.

[44]:

```
queryResult = spark.sql("""
    SELECT customerID, Contract, MonthlyCharges
    FROM (
        SELECT customerID, Contract, MonthlyCharges,
        DENSE_RANK(MonthlyCharges) OVER(PARTITION BY Contract ORDER BY␣
 ↪MonthlyCharges DESC) AS rank
        FROM churnData
    ) AS _
    WHERE rank<=5
""")
queryResult.show()


# below table shows top 5 customers within each contract type to pay highest␣
 ↪Monthly charges
```

```
+----------+--------------+--------------+
|customerID|      Contract|MonthlyCharges|
+----------+--------------+--------------+
|2302-ANTDP|Month-to-month|        117.45|
|8016-NCFVO|Month-to-month|         116.5|
|9659-QEQSY|Month-to-month|        115.65|
|4361-BKAXE|Month-to-month|         114.5|
|6710-HSJRD|Month-to-month|         114.1|
|5734-EJKXG|      One year|         118.6|
|8199-ZLLSA|      One year|        118.35|
|2889-FPWRM|      One year|         117.8|
|4282-MSACW|      One year|         117.2|
|3680-CTHUH|      One year|         116.6|
|7569-NMZYQ|      Two year|        118.75|
|8984-HPEMB|      Two year|        118.65|
|5989-AXPUC|      Two year|         118.6|
|9924-JPRMC|      Two year|         118.2|
|3810-DVDQQ|      Two year|         117.6|
+----------+--------------+--------------+
```

[74]:

```
# queryDF = queryResult.toPandas()

# # ???
# plt.figure(figsize=(12, 8))
# sns.barplot(data=queryDF, x='Contract', y='MonthlyCharges', hue='customerID')
# plt.title('MonthlyCharges by Contract')
# plt.xlabel('Contract')
```

```
# plt.ylabel('MonthlyCharges')
# plt.show()
```

0.0.23 w) Calculate the churn rate in each Contract type and rank the contracts by churn rate.

[46]:
```
spark.sql("""
    SELECT *, DENSE_RANK() OVER(ORDER BY churn_rate) AS rank
    FROM (
        SELECT Contract,
               round(SUM(CASE WHEN Churn="Yes" THEN 1 ELSE 0 END)*100/COUNT(*)␣
 ↪,3) AS churn_rate
        FROM churnData
        GROUP BY Contract
    ) AS _
""").show()


# INSIGHT: customers with longer contract period have much lower churn rate
# when compared to those with shorter contract period
```

```
+--------------+----------+----+
|      Contract|churn_rate|rank|
+--------------+----------+----+
|      Two year|     2.849|   1|
|      One year|    11.277|   2|
|Month-to-month|     42.71|   3|
+--------------+----------+----+
```

0.0.24 x) Perform an in-depth analysis of customers using window functions to understand customer rankings, distribution, and trends in charges and tenure.

[78]:
```
# ranking customers by decreasing order of monthly charges and tenure
spark.sql("""
    SELECT customerID,
           MonthlyCharges, DENSE_RANK() OVER (ORDER BY MonthlyCharges DESC) AS␣
 ↪MonthlyChargesRank,
           Tenure, DENSE_RANK() OVER (ORDER BY Tenure DESC) AS TenureRank
    FROM churnData
    ORDER BY MonthlyCharges DESC, Tenure DESC
""").show()
```

```
# INSIGHT: from the data, we can find that generally, customers with highest␣
↪monthly charges
# also have the longest tenures
```

```
+----------+--------------+------------------+------+----------+
|customerID|MonthlyCharges|MonthlyChargesRank|Tenure|TenureRank|
+----------+--------------+------------------+------+----------+
|7569-NMZYQ|        118.75|                 1|    72|         1|
|8984-HPEMB|        118.65|                 2|    71|         2|
|5989-AXPUC|         118.6|                 3|    68|         5|
|5734-EJKXG|         118.6|                 3|    61|        12|
|8199-ZLLSA|        118.35|                 4|    67|         6|
|9924-JPRMC|         118.2|                 5|    72|         1|
|2889-FPWRM|         117.8|                 6|    72|         1|
|3810-DVDQQ|         117.6|                 7|    72|         1|
|9739-JLPQJ|         117.5|                 8|    72|         1|
|2302-ANTDP|        117.45|                 9|    48|        25|
|6904-JLBGY|        117.35|                10|    72|         1|
|4282-MSACW|         117.2|                11|    68|         5|
|6650-BWFRT|        117.15|                12|    72|         1|
|9788-HNGUT|        116.95|                13|    72|         1|
|1488-PBLJN|        116.85|                14|    72|         1|
|0017-IUDMW|         116.8|                15|    72|         1|
|8628-MFKAX|        116.75|                16|    72|         1|
|3258-ZKPAI|         116.6|                17|    72|         1|
|3680-CTHUH|         116.6|                17|    60|        13|
|3795-CAWEX|        116.55|                18|    70|         3|
+----------+--------------+------------------+------+----------+
only showing top 20 rows
```

[80]:

```
# comparing average of monthly and total charges with tenure
spark.sql("""
    SELECT tenure,
           round(AVG(MonthlyCharges), 3) AS avg_MonthlyCharges,
           round(AVG(TotalCharges), 3) AS avg_TotalCharges,
           COUNT(*) AS count
    FROM churnData
    GROUP BY Tenure
    ORDER BY Tenure
""").show()


# INSIGHT: the trend is that average monthly and total charges increase as␣
↪tenure also increases
# ie, they are directly related to the tenure
```

```
+------+------------------+----------------+-----+
|tenure|avg_MonthlyCharges|avg_TotalCharges|count|
+------+------------------+----------------+-----+
|     1|            50.486|          50.486|  613|
|     2|            57.206|         114.332|  238|
|     3|            58.015|          174.69|  200|
|     4|            57.433|         230.531|  176|
|     5|            61.004|         304.491|  133|
|     6|            56.589|         336.175|  110|
|     7|            59.642|         418.391|  131|
|     8|            57.245|         462.789|  123|
|     9|            62.565|         564.141|  119|
|    10|            58.825|         593.735|  116|
|    11|            58.473|         648.084|   99|
|    12|            56.836|         671.849|  117|
|    13|            60.071|         777.529|  109|
|    14|            62.666|          873.03|   76|
|    15|            60.592|         910.693|   99|
|    16|            62.546|        1002.912|   80|
|    17|            62.655|        1059.293|   87|
|    18|             59.73|        1072.656|   97|
|    19|             57.87|        1091.256|   73|
|    20|            60.713|         1225.82|   71|
+------+------------------+----------------+-----+
only showing top 20 rows
```

[49]:

```
# find cumulative distribution of monthly charges, partitoned by tenure
spark.sql("""
    SELECT customerID, tenure, MonthlyCharges,
           CUME_DIST() OVER (PARTITION BY tenure ORDER BY MonthlyCharges) as␣
 ↪cume_dist
    FROM churnData
    ORDER BY tenure, cume_dist
""").show()
```

```
+----------+------+--------------+--------------------+
|customerID|tenure|MonthlyCharges|           cume_dist|
+----------+------+--------------+--------------------+
|2967-MXRAV|     1|          18.8|0.001631321370309…|
|8992-CEUEN|     1|         18.85|0.004893964110929853|
|9318-NKNFC|     1|         18.85|0.004893964110929853|
|9975-SKRNR|     1|          18.9|0.006525285481239…|
|1423-BMPBQ|     1|          19.0|0.008156606851549755|
|1015-OWJKI|     1|         19.05|0.009787928221859706|
|6121-VZNQB|     1|          19.1| 0.01468189233278956|
|9441-QHEVC|     1|          19.1| 0.01468189233278956|
|6569-KTMDU|     1|          19.1| 0.01468189233278956|
```

```
|7302-ZHMHP|     1|         19.15| 0.01631321370309951|
|3308-MHOOC|     1|          19.2|0.022838499184339316|
|3373-YZZYM|     1|          19.2|0.022838499184339316|
|1663-MHLHE|     1|          19.2|0.022838499184339316|
|4232-JGKIY|     1|          19.2|0.022838499184339316|
|5510-BOIUJ|     1|         19.25| 0.02773246329526917|
|9374-YOLBJ|     1|         19.25| 0.02773246329526917|
|1098-TDVUQ|     1|         19.25| 0.02773246329526917|
|4667-OHGKG|     1|          19.3| 0.03425774877650897|
|1724-IQWNM|     1|          19.3| 0.03425774877650897|
|7926-IJOOU|     1|          19.3| 0.03425774877650897|
+----------+------+--------------+--------------------+
only showing top 20 rows
```

[91]:

```
# cumulative sum of monthly charges and tenure

spark.sql("""
    SELECT customerID, tenure, MonthlyCharges,
           ROUND(SUM(MonthlyCharges)
               OVER (ORDER BY tenure ROWS BETWEEN UNBOUNDED PRECEDING AND␣
 ↪CURRENT ROW), 5) AS monthlyCharges_movingSum
    FROM churnData
    ORDER BY tenure
""").show()
```

```
+----------+------+--------------+------------------------+
|customerID|tenure|MonthlyCharges|monthlyCharges_movingAvg|
+----------+------+--------------+------------------------+
|7590-VHVEG|     1|         29.85|                   29.85|
|8779-QRDMV|     1|         39.65|                    69.5|
|1066-JKSGK|     1|         20.15|                   89.65|
|8665-UTDHZ|     1|          30.2|                  119.85|
|7310-EGVHZ|     1|          20.2|                  140.05|
|3413-BMNZE|     1|         45.25|                   185.3|
|2273-QCKXA|     1|         49.05|                  234.35|
|5919-TMRGD|     1|         79.35|                   313.7|
|2424-WVHPL|     1|          74.7|                   388.4|
|6380-ARCEH|     1|          20.2|                   408.6|
|3679-XASPY|     1|         19.45|                  428.05|
|3930-ZGWVE|     1|         19.75|                   447.8|
|3091-FYHKI|     1|         35.45|                  483.25|
|0390-DCFDQ|     1|         70.45|                   553.7|
|2135-RXIHG|     1|         45.65|                  599.35|
|6317-YPKDH|     1|         29.95|                   629.3|
|6582-OIVSP|     1|          45.3|                   674.6|
|1024-GUALD|     1|          24.8|                   699.4|
|3645-DEYGF|     1|         20.75|                  720.15|
```

```
|1285-OKIPP|     1|          79.9|                  800.05|
+----------+------+--------------+------------------------+
only showing top 20 rows
```

[92]:
```python
# moving average of monthly charges over tenure

spark.sql("""
    SELECT customerID, tenure, MonthlyCharges,
           ROUND(AVG(MonthlyCharges) OVER (ORDER BY tenure ROWS BETWEEN␣
 ↪UNBOUNDED PRECEDING AND CURRENT ROW), 5) AS monthlyCharges_movingAvg
    FROM churnData
    ORDER BY tenure
""").show()
```

```
+----------+------+--------------+------------------------+
|customerID|tenure|MonthlyCharges|monthlyCharges_movingAvg|
+----------+------+--------------+------------------------+
|7590-VHVEG|     1|         29.85|                   29.85|
|8779-QRDMV|     1|         39.65|                   34.75|
|1066-JKSGK|     1|         20.15|                29.88333|
|8665-UTDHZ|     1|          30.2|                 29.9625|
|7310-EGVHZ|     1|          20.2|                   28.01|
|3413-BMNZE|     1|         45.25|                30.88333|
|2273-QCKXA|     1|         49.05|                33.47857|
|5919-TMRGD|     1|         79.35|                 39.2125|
|2424-WVHPL|     1|          74.7|                43.15556|
|6380-ARCEH|     1|          20.2|                   40.86|
|3679-XASPY|     1|         19.45|                38.91364|
|3930-ZGWVE|     1|         19.75|                37.31667|
|3091-FYHKI|     1|         35.45|                37.17308|
|0390-DCFDQ|     1|         70.45|                   39.55|
|2135-RXIHG|     1|         45.65|                39.95667|
|6317-YPKDH|     1|         29.95|                39.33125|
|6582-OIVSP|     1|          45.3|                39.68235|
|1024-GUALD|     1|          24.8|                38.85556|
|3645-DEYGF|     1|         20.75|                37.90263|
|1285-OKIPP|     1|          79.9|                 40.0025|
+----------+------+--------------+------------------------+
only showing top 20 rows
```

[ ]: