In [1]:
```
import findspark
findspark.init()
```

In [2]:
```
import pyspark
```

In [3]:
```
from pyspark.sql import SparkSession
```

In [4]:
```
spark = SparkSession.builder.appName('BigdataProject').getOrCreate()
```

In [5]:
```
spark
```

Out[5]:

SparkSession - in-memory

SparkContext

Spark UI

Version	`v3.3.1`
Master	`local[*]`
AppName	`BigdataProject`

In [6]:
```
df = spark.read.option('header','true').csv('CIS_Automotive_Kaggle_Sample.csv',inferSchema=True)
```

In [7]:
```
df.columns
```

```
Out[7]: ['vin',
          'stockNum',
          'firstSeen',
          'lastSeen',
          'msrp',
          'askPrice',
          'mileage',
          'isNew',
          'color',
          'interiorColor',
          'brandName',
          'modelName',
          'dealerID',
          'vf_ABS',
          'vf_ActiveSafetySysNote',
          'vf_AdaptiveCruiseControl',
          'vf_AdaptiveDrivingBeam',
          'vf_AdaptiveHeadlights',
          'vf_AdditionalErrorText',
          'vf_AirBagLocCurtain',
          'vf_AirBagLocFront',
          'vf_AirBagLocKnee',
          'vf_AirBagLocSeatCushion',
          'vf_AirBagLocSide',
          'vf_AutoReverseSystem',
          'vf_AutomaticPedestrianAlertingSound',
          'vf_AxleConfiguration',
          'vf_Axles',
          'vf_BasePrice',
          'vf_BatteryA',
          'vf_BatteryA_to',
          'vf_BatteryCells',
          'vf_BatteryInfo',
          'vf_BatteryKWh',
          'vf_BatteryKWh_to',
          'vf_BatteryModules',
          'vf_BatteryPacks',
          'vf_BatteryType',
          'vf_BatteryV',
          'vf_BatteryV_to',
          'vf_BedLengthIN',
          'vf_BedType',
          'vf_BlindSpotMon',
          'vf_BodyCabType',
          'vf_BodyClass',
          'vf_BrakeSystemDesc',
          'vf_BrakeSystemType',
          'vf_BusFloorConfigType',
          'vf_BusLength',
          'vf_BusType',
          'vf_CAN_AACN',
          'vf_CIB',
          'vf_CashForClunkers',
          'vf_ChargerLevel',
          'vf_ChargerPowerKW',
          'vf_CoolingType',
          'vf_CurbWeightLB',
          'vf_CustomMotorcycleType',
          'vf_DaytimeRunningLight',
          'vf_DestinationMarket',
          'vf_DisplacementCC',
          'vf_DisplacementCI',
          'vf_DisplacementL',
          'vf_Doors',
          'vf_DriveType',
          'vf_DriverAssist',
          'vf_DynamicBrakeSupport',
          'vf_EDR',
          'vf_ESC',
          'vf_EVDriveUnit',
          'vf_ElectrificationLevel',
          'vf_EngineConfiguration',
          'vf_EngineCycles',
          'vf_EngineCylinders',
          'vf_EngineHP',
          'vf_EngineHP_to',
          'vf_EngineKW',
          'vf_EngineManufacturer',
          'vf_EngineModel',
          'vf_EntertainmentSystem',
          'vf_ForwardCollisionWarning',
          'vf_FuelInjectionType',
          'vf_FuelTypePrimary',
          'vf_FuelTypeSecondary',
          'vf_GCWR',
          'vf_GCWR_to',
          'vf_GVWR',
          'vf_GVWR_to',
          'vf_KeylessIgnition',
          'vf_LaneDepartureWarning',
          'vf_LaneKeepSystem',
          'vf_LowerBeamHeadlampLightSource',
          'vf_Make',
          'vf_MakeID',
          'vf_Manufacturer',
          'vf_ManufacturerId',
          'vf_Model',
          'vf_ModelID',
          'vf_ModelYear',
          'vf_MotorcycleChassisType',
          'vf_MotorcycleSuspensionType',
          'vf_NCSABodyType',
          'vf_NCSAMake',
          'vf_NCSAMapExcApprovedBy',
          'vf_NCSAMapExcApprovedOn',
          'vf_NCSAMappingException',
          'vf_NCSAModel',
          'vf_NCSANote',
          'vf_Note',
          'vf_OtherBusInfo',
          'vf_OtherEngineInfo',
          'vf_OtherMotorcycleInfo',
          'vf_OtherRestraintSystemInfo',
          'vf_OtherTrailerInfo',
          'vf_ParkAssist',
          'vf_PedestrianAutomaticEmergencyBraking',
          'vf_PlantCity',
          'vf_PlantCompanyName',
          'vf_PlantCountry',
          'vf_PlantState',
          'vf_PossibleValues',
          'vf_Pretensioner',
          'vf_RearCrossTrafficAlert',
          'vf_RearVisibilitySystem',
          'vf_SAEAutomationLevel',
          'vf_SAEAutomationLevel_to',
          'vf_SeatBeltsAll',
          'vf_SeatRows',
          'vf_Seats',
          'vf_SemiautomaticHeadlampBeamSwitching',
          'vf_Series',
          'vf_Series2',
          'vf_SteeringLocation',
          'vf_SuggestedVIN',
          'vf_TPMS',
          'vf_TopSpeedMPH',
          'vf_TrackWidth',
          'vf_TractionControl',
          'vf_TrailerBodyType',
          'vf_TrailerLength',
          'vf_TrailerType',
          'vf_TransmissionSpeeds',
          'vf_TransmissionStyle',
          'vf_Trim',
          'vf_Trim2',
          'vf_Turbo',
          'vf_VIN',
          'vf_ValveTrainDesign',
          'vf_VehicleType',
          'vf_WheelBaseLong',
          'vf_WheelBaseShort',
          'vf_WheelBaseType',
          'vf_WheelSizeFront',
          'vf_WheelSizeRear',
          'vf_Wheels',
          'vf_Windows']
```

In [8]:
```
from pyspark.sql.types import StringType, BooleanType, IntegerType
```

In [9]:
```
df_pyspark = df.select(['vin','firstseen','lastseen','askPrice','mileage','isNew','brandName','modelNar
  'vf_AirBagLocFront','vf_AirBagLocKnee', 'vf_AirBagLocSide','vf_Axles','vf_BasePrice','vf_Displacement
  'vf_EngineCylinders','vf_EngineHP','vf_EngineKW','vf_SeatRows','vf_Seats','vf_TopSpeedMPH','vf_Transr
```

In [10]:
```
df_pyspark
```

Out[10]:
```
DataFrame[vin: string, firstseen: timestamp, lastseen: timestamp, askPrice: int, mileage: int, isNew:
boolean, brandName: string, modelName: string, vf_AirBagLocFront: string, vf_AirBagLocKnee: string, vf
_AirBagLocSide: string, vf_Axles: int, vf_BasePrice: double, vf_DisplacementCC: double, vf_Displacemen
tCI: double, vf_DisplacementL: double, vf_Doors: int, vf_EngineCylinders: int, vf_EngineHP: double, vf
_EngineKW: double, vf_SeatRows: int, vf_Seats: int, vf_TopSpeedMPH: int, vf_TransmissionSpeeds: int, v
f_WheelBaseShort: double, vf_WheelSizeFront: int, vf_WheelSizeRear: int, vf_Wheels: int, vf_Windows: i
nt, msrp: int]
```

In [11]:
```
#Q1
from pyspark.sql.functions import datediff,col,lit
df1 = df_pyspark.withColumn('Time in lot',datediff(df['lastSeen'],df['FirstSeen']))
```

In [12]:
```
#dropping first seen and last seen columns as we have already added a new column 'Time in Lot' for it
df2 = df1.drop(col("firstseen"))
df2 = df2.drop(col("lastseen"))
df2.dtypes
```

Out[12]:
```
[('vin', 'string'),
 ('askPrice', 'int'),
 ('mileage', 'int'),
 ('isNew', 'boolean'),
 ('brandName', 'string'),
 ('modelName', 'string'),
 ('vf_AirBagLocFront', 'string'),
 ('vf_AirBagLocKnee', 'string'),
 ('vf_AirBagLocSide', 'string'),
 ('vf_Axles', 'int'),
 ('vf_BasePrice', 'double'),
 ('vf_DisplacementCC', 'double'),
 ('vf_DisplacementCI', 'double'),
 ('vf_DisplacementL', 'double'),
 ('vf_Doors', 'int'),
 ('vf_EngineCylinders', 'int'),
 ('vf_EngineHP', 'double'),
 ('vf_EngineKW', 'double'),
 ('vf_SeatRows', 'int'),
 ('vf_Seats', 'int'),
 ('vf_TopSpeedMPH', 'int'),
 ('vf_TransmissionSpeeds', 'int'),
 ('vf_WheelBaseShort', 'double'),
 ('vf_WheelSizeFront', 'int'),
 ('vf_WheelSizeRear', 'int'),
 ('vf_Wheels', 'int'),
 ('vf_Windows', 'int'),
 ('msrp', 'int'),
 ('Time in lot', 'int')]
```

In [13]:
```
from pyspark.sql.functions import mean
df2 = df2.na.drop(subset=['msrp','vf_BasePrice','askPrice'])
```

In [14]:
```
df2.select('msrp','vf_BasePrice','askPrice').show()
```

```
+-----+------------+--------+
| msrp|vf_BasePrice|askPrice|
+-----+------------+--------+
|12387|     23475.0|   12387|
|12970|     26500.0|   12970|
|15218|     23940.0|   15218|
|18755|     38495.0|   18755|
|36999|     34175.0|   36999|
|18276|     24105.0|   18276|
|22140|     28000.0|   22140|
|    0|     34175.0|       0|
|20494|     30530.0|   20494|
|19026|     26500.0|   19026|
|18951|     26500.0|   18951|
|16649|     27995.0|   16649|
|16671|     30420.0|   16671|
|16998|     27995.0|   16998|
|20294|     30530.0|   20294|
|45996|     63000.0|   45996|
|23537|     29995.0|   23537|
|12999|     23740.0|   12999|
|13398|     23740.0|   13398|
|14121|     19995.0|   13692|
+-----+------------+--------+
only showing top 20 rows
```

In [15]:

```
import pyspark.sql.functions as F
df3 = df2.withColumn('isNew', F.when(df2['isNew'] == 'FALSE', 0).otherwise(1))
df3.select('isNew').show()
```

```
+-----+
|isNew|
+-----+
|    0|
|    0|
|    0|
|    0|
|    0|
|    1|
|    1|
|    1|
|    0|
|    1|
|    1|
|    0|
|    0|
|    0|
|    0|
|    0|
|    1|
|    0|
|    0|
|    0|
+-----+
only showing top 20 rows
```

In [16]:

```
#changing vf_AirBagLocFront from string to integer
from pyspark.sql.functions import when
df4 = df3.withColumn('vf_AirBagLocFront', (when(df3['vf_AirBagLocFront'] == '1st Row (Driver & Passenge
df4.select('vf_AirBagLocFront').show(45)
```

```
+-----------------+
|vf_AirBagLocFront|
+-----------------+
|                2|
|                2|
|                2|
|                2|
|                2|
|                2|
|                2|
|                2|
|                2|
|                2|
|                2|
|                2|
|                2|
|                2|
|                2|
|                2|
|                2|
|                2|
|                2|
|                2|
|                2|
|                2|
|                2|
|                2|
|                2|
|                2|
|                2|
|                2|
|                2|
|                2|
|                2|
|                2|
|                2|
|                2|
|                2|
|                2|
|                2|
|                2|
|                2|
|                2|
|                2|
|                2|
|                2|
|                2|
|                2|
+-----------------+
only showing top 45 rows
```

In [17]:

```
#changing vf_AirBagLocSide from string to integer
df5 = df4.withColumn('vf_AirBagLocSide', (when(df4['vf_AirBagLocSide'] == '1st Row (Driver & Passenger
df5.select('vf_AirBagLocSide').show()
```

```
+----------------+
|vf_AirBagLocSide|
+----------------+
|               4|
|               4|
|               4|
|               6|
|               6|
|               4|
|               4|
|               6|
|               6|
|               4|
|               4|
|               4|
|               4|
|               4|
|               6|
|               2|
|               4|
|               4|
|               4|
|               4|
+----------------+
only showing top 20 rows
```

In [18]:
```
#changing vf_AirBagLocKnee from string to integer
df6 = df5.withColumn('vf_AirBagLocKnee', (when(df5['vf_AirBagLocKnee'] == '1st Row (Driver & Passenger)
df6.select('vf_AirBagLocKnee').show()
```

```
+----------------+
|vf_AirBagLocKnee|
+----------------+
|               2|
|               2|
|               0|
|               0|
|               0|
|               0|
|               0|
|               0|
|               0|
|               0|
|               0|
|               0|
|               2|
|               0|
|               0|
|               0|
|               0|
|               2|
|               2|
|               0|
+----------------+
only showing top 20 rows
```

In [19]:
```
df6.select('vf_Axles').show()
```

```
+--------+
|vf_Axles|
+--------+
|       2|
|       2|
|       2|
|       2|
|       2|
|       2|
|       2|
|       2|
|       2|
|       2|
|       2|
|       2|
|       2|
|       2|
|       2|
|       2|
|       2|
|       2|
|       2|
|       2|
+--------+
only showing top 20 rows
```

In [20]:
```
#vf_Axles - filling null values with average
avg_axles = df6.agg({'vf_Axles': 'mean' })
df7 = df6.na.fill(value=avg_axles.first()[0],subset=["vf_Axles"])
df7.select('vf_Axles').show()
```

```
+--------+
|vf_Axles|
+--------+
|       2|
|       2|
|       2|
|       2|
|       2|
|       2|
|       2|
|       2|
|       2|
|       2|
|       2|
|       2|
|       2|
|       2|
|       2|
|       2|
|       2|
|       2|
|       2|
|       2|
+--------+
only showing top 20 rows
```

In [21]:
```
df7.select('vf_BasePrice').show()
```

```
+------------+
|vf_BasePrice|
+------------+
|     23475.0|
|     26500.0|
|     23940.0|
|     38495.0|
|     34175.0|
|     24105.0|
|     28000.0|
|     34175.0|
|     30530.0|
|     26500.0|
|     26500.0|
|     27995.0|
|     30420.0|
|     27995.0|
|     30530.0|
|     63000.0|
|     29995.0|
|     23740.0|
|     23740.0|
|     19995.0|
+------------+
only showing top 20 rows
```

In [22]:

```
#vf_BasePrice - filling null values with average
avg_bprice = df7.agg({'vf_BasePrice': 'mean' })
df8 = df7.na.fill(value=int(avg_bprice.first()[0]),subset=["vf_BasePrice"])
df8.select('vf_BasePrice').show()
```

```
+------------+
|vf_BasePrice|
+------------+
|     23475.0|
|     26500.0|
|     23940.0|
|     38495.0|
|     34175.0|
|     24105.0|
|     28000.0|
|     34175.0|
|     30530.0|
|     26500.0|
|     26500.0|
|     27995.0|
|     30420.0|
|     27995.0|
|     30530.0|
|     63000.0|
|     29995.0|
|     23740.0|
|     23740.0|
|     19995.0|
+------------+
only showing top 20 rows
```

In [23]:

```
#vf_SeatRows, vf_Seats, vf_TopSpeedMPH, vf_TransmissionSpeeds, vf_WheelBaseShort, vf_WheelSizeFront,
#vf_WheelSizeRear, vf_Wheels, vf_Windows - filling null values with average
avg_dcc = df8.agg({'vf_DisplacementCC': 'mean' })
df9 = df8.na.fill(value=int(avg_dcc.first()[0]),subset=["vf_DisplacementCC"])
avg_dci = df9.agg({'vf_DisplacementCI': 'mean' })
df9 = df9.na.fill(value=int(avg_dci.first()[0]),subset=["vf_DisplacementCI"])
avg_dl = df9.agg({'vf_DisplacementL': 'mean' })
df9 = df9.na.fill(value=int(avg_dl.first()[0]),subset=["vf_DisplacementL"])
df9 = df9.na.fill(value=4,subset=['vf_Doors'])
avg_cyl = df9.agg({'vf_EngineCylinders': 'mean' })
df9 = df9.na.fill(value=int(avg_cyl.first()[0]),subset=["vf_EngineCylinders"])
avg_ehp = df9.agg({'vf_EngineHP': 'mean' })
df9 = df9.na.fill(value=avg_ehp.first()[0],subset=["vf_EngineHP"])
avg_ekw = df9.agg({'vf_EngineKW': 'mean' })
df9 = df9.na.fill(value=avg_ekw.first()[0],subset=["vf_EngineKW"])
avg_sr = df9.agg({'vf_SeatRows': 'mean' })
```

```
df9 = df9.na.fill(value=int(avg_sr.first()[0]),subset=["vf_SeatRows"])
avg_seats = df9.agg({'vf_Seats': 'mean' })
df9 = df9.na.fill(value=int(avg_seats.first()[0]),subset=["vf_Seats"])
avg_speed = df9.agg({'vf_TopSpeedMPH': 'mean' })
df9 = df9.na.fill(value=int(avg_speed.first()[0]),subset=["vf_TopSpeedMPH"])
avg_tspeed = df9.agg({'vf_TransmissionSpeeds': 'mean' })
df9 = df9.na.fill(value=int(avg_tspeed.first()[0]),subset=["vf_TransmissionSpeeds"])
avg_wbs = df9.agg({'vf_WheelBaseShort': 'mean' })
df9 = df9.na.fill(value=avg_wbs.first()[0],subset=["vf_WheelBaseShort"])
avg_wsf = df9.agg({'vf_WheelSizeFront': 'mean' })
df9 = df9.na.fill(value=int(avg_wsf.first()[0]),subset=["vf_WheelSizeFront"])
avg_wsr = df9.agg({'vf_WheelSizeRear': 'mean' })
df9 = df9.na.fill(value=int(avg_wsr.first()[0]),subset=["vf_WheelSizeRear"])
avg_wheels = df9.agg({'vf_Wheels': 'mean' })
df9 = df9.na.fill(value=int(avg_wheels.first()[0]),subset=["vf_Wheels"])
df9 = df9.na.fill(value=4,subset=['vf_Windows'])
df9.select('vf_DisplacementCC','vf_DisplacementCI','vf_DisplacementL','vf_Doors','vf_EngineCylinders',
+-----------------+-----------------+----------------+--------+------------------+------------------+-
----------------+-----------+--------+--------------+---------------------+-----------------+---------
--------+----------------+---------+----------+
|vf_DisplacementCC|vf_DisplacementCI|vf_DisplacementL|vf_Doors|vf_EngineCylinders|       vf_EngineHP|
vf_EngineKW|vf_SeatRows|vf_Seats|vf_TopSpeedMPH|vf_TransmissionSpeeds|vf_WheelBaseShort|vf_WheelSizeFr
ont|vf_WheelSizeRear|vf_Wheels|vf_Windows|
+-----------------+-----------------+----------------+--------+------------------+------------------+-
----------------+-----------+--------+--------------+---------------------+-----------------+---------
--------+----------------+---------+----------+
|           1400.0|   85.43324173262|             1.4|       4|                 4|231.21825842998672|1
72.3428422091031|          2|       5|           130|                    6|            106.3|
17|              17|        4|         4|
|           2000.0|  122.04748818946|             2.0|       4|                 4|             188.0|
140.1916|          2|       5|           105|                    6|            112.2|               17
|              17|        4|         4|
|           2500.0|  152.55936023683|             2.5|       4|                 4|             171.0|
127.5147|          2|       5|           114|                    6|            105.9|               17
|              17|        4|         4|
|           3500.0|  213.58310433156|             3.5|       4|                 6|             287.0|
214.0159|          2|       7|           126|                    6|            117.9|               19
|              19|        4|         4|
|           3500.0|  213.58310433156|             3.5|       4|                 6|             290.0|
216.253|          3|       7|           122|                    6|            112.8|               18|
18|        4|         4|
|           2500.0|  152.55936023683|             2.5|       4|                 4|             171.0|
127.5147|          2|       5|           126|                    6|            105.9|               17
|              17|        4|         4|
|           1500.0|   91.53561614209|             1.5|       4|                 4|             170.0|
126.769|          2|       5|           126|                    6|            105.9|               17|
17|        4|         4|
|           3500.0|  213.58310433156|             3.5|       4|                 6|             290.0|
216.253|          3|       7|           122|                    6|            112.8|               18|
18|        4|         4|
|           2500.0|  152.55936023683|             2.5|       4|                 4|             169.0|
126.0233|          3|       7|           126|                    6|            120.6|               16
|              16|        4|         4|
|           1500.0|   91.53561614209|             1.5|       4|                 4|             170.0|
126.769|          2|       5|           126|                    6|            105.9|               17|
17|        4|         4|
|           1500.0|   91.53561614209|             1.5|       4|                 4|             170.0|
126.769|          2|       5|           126|                    6|            105.9|               17|
17|        4|         4|
|           3600.0|  219.68547874103|             3.6|       4|                 6|             283.0|
211.0331|          2|       7|           126|                    4|            113.8|               19
|              19|        4|         4|
|           2500.0|  152.55936023683|             2.5|       4|                 4|231.21825842998672|1
72.3428422091031|          2|       5|           126|                    6|            111.7|
18|              18|        4|         4|
|           3600.0|  219.68547874103|             3.6|       4|                 6|             283.0|
211.0331|          2|       7|           126|                    4|            113.8|               19
|              19|        4|         4|
|           2500.0|  152.55936023683|             2.5|       4|                 4|             169.0|
126.0233|          3|       7|           126|                    6|            120.6|               16
|              16|        4|         4|
|           3000.0|   183.0712322841|             3.0|       4|                 6|             340.0|
253.538|          1|       2|           126|                    8|            103.2|               18|
18|        4|         4|
|           2000.0|  122.04748818946|             2.0|       4|                 4|             245.0|
182.6965|          2|       5|           126|                    6|            112.2|               18
|              18|        4|         4|
|           1500.0|   91.53561614209|             1.5|       4|                 4|             181.0|
134.9717|          2|       5|           155|                    6|            112.2|               17
|              17|        4|         4|
|           1500.0|   91.53561614209|             1.5|       4|                 4|             181.0|
134.9717|          2|       5|           155|                    6|            112.2|               17
|              17|        4|         4|
|           2400.0|  146.45698582735|             2.4|       4|                 4|             180.0|
134.226|          2|       5|           126|                    6|            101.2|               16|
16|        4|         4|
+-----------------+-----------------+----------------+--------+------------------+------------------+-
----------------+-----------+--------+--------------+---------------------+-----------------+---------
--------+----------------+---------+----------+
only showing top 20 rows
```

In [24]:

```
df9.dtypes
```

Out[24]:

```
[('vin', 'string'),
 ('askPrice', 'int'),
 ('mileage', 'int'),
 ('isNew', 'int'),
 ('brandName', 'string'),
 ('modelName', 'string'),
 ('vf_AirBagLocFront', 'int'),
 ('vf_AirBagLocKnee', 'int'),
 ('vf_AirBagLocSide', 'int'),
 ('vf_Axles', 'int'),
 ('vf_BasePrice', 'double'),
 ('vf_DisplacementCC', 'double'),
 ('vf_DisplacementCI', 'double'),
 ('vf_DisplacementL', 'double'),
 ('vf_Doors', 'int'),
 ('vf_EngineCylinders', 'int'),
 ('vf_EngineHP', 'double'),
 ('vf_EngineKW', 'double'),
 ('vf_SeatRows', 'int'),
 ('vf_Seats', 'int'),
 ('vf_TopSpeedMPH', 'int'),
 ('vf_TransmissionSpeeds', 'int'),
 ('vf_WheelBaseShort', 'double'),
 ('vf_WheelSizeFront', 'int'),
 ('vf_WheelSizeRear', 'int'),
 ('vf_Wheels', 'int'),
 ('vf_Windows', 'int'),
 ('msrp', 'int'),
 ('Time in lot', 'int')]
```

In [25]:

```
from pyspark.ml.feature import VectorAssembler
ip_cols = ['askPrice','mileage','isNew','vf_AirBagLocFront','vf_AirBagLocKnee', 'vf_AirBagLocSide','vf_
  'vf_EngineCylinders','vf_EngineHP','vf_EngineKW','vf_SeatRows','vf_Seats','vf_TopSpeedMPH','vf_Transr
op_col = "Features"
vec_df = VectorAssembler(inputCols = ip_cols, outputCol = op_col)

df_ml = vec_df.transform(df9)
df_ml.select(['Features']).toPandas().head(5)
```

Out[25]:

	Features
0	[12387.0, 0.0, 0.0, 2.0, 2.0, 4.0, 2.0, 23475....
1	[12970.0, 0.0, 0.0, 2.0, 2.0, 4.0, 2.0, 26500....
2	[15218.0, 0.0, 0.0, 2.0, 0.0, 4.0, 2.0, 23940....
3	[18755.0, 0.0, 0.0, 2.0, 0.0, 6.0, 2.0, 38495....
4	[36999.0, 0.0, 0.0, 2.0, 0.0, 6.0, 2.0, 34175....

In [26]:

```
final_df = df_ml.select(['Features','msrp'])
final_df.show(1)
```

```
+--------------------+-----+
|            Features| msrp|
+--------------------+-----+
|[12387.0,0.0,0.0,...|12387|
+--------------------+-----+
only showing top 1 row
```

In [27]:

```
len_df = final_df.count()

train_len = int(0.7*len_df)
train_len
```

Out[27]:

```
1299720
```

In [28]:

```
test_df = final_df
train_df = final_df.limit(train_len)
train_df.show()
train_df.count()
```

```
+--------------------+-----+
|            Features| msrp|
+--------------------+-----+
|[12387.0,0.0,0.0,...|12387|
|[12970.0,0.0,0.0,...|12970|
|[15218.0,0.0,0.0,...|15218|
|[18755.0,0.0,0.0,...|18755|
|[36999.0,0.0,0.0,...|36999|
|[18276.0,0.0,1.0,...|18276|
|[22140.0,0.0,1.0,...|22140|
|[0.0,0.0,1.0,2.0,...|    0|
|[20494.0,0.0,0.0,...|20494|
|[19026.0,0.0,1.0,...|19026|
|[18951.0,0.0,1.0,...|18951|
|[16649.0,0.0,0.0,...|16649|
|[16671.0,0.0,0.0,...|16671|
|[16998.0,0.0,0.0,...|16998|
|[20294.0,0.0,0.0,...|20294|
|[45996.0,0.0,0.0,...|45996|
|[23537.0,0.0,1.0,...|23537|
|[12999.0,0.0,0.0,...|12999|
|[13398.0,0.0,0.0,...|13398|
|[13692.0,0.0,0.0,...|14121|
+--------------------+-----+
only showing top 20 rows
```

Out[28]:

```
1299720
```

In [29]:

```
test_df = test_df.subtract(train_df)
test_df.show()
test_df.count()
```

```
+--------------------+-----+
|            Features| msrp|
+--------------------+-----+
|[0.0,0.0,1.0,2.0,...|    0|
|[0.0,0.0,1.0,2.0,...|    0|
|[0.0,0.0,1.0,2.0,...|    0|
|[0.0,0.0,1.0,2.0,...|    0|
|[0.0,0.0,1.0,2.0,...|    0|
|[0.0,3.0,1.0,2.0,...|    0|
|[0.0,9.0,1.0,2.0,...|    0|
|[0.0,12153.0,0.0,...|    0|
|[0.0,15144.0,0.0,...|    0|
|[11499.0,45603.0,...|12399|
|[12928.0,0.0,1.0,...|16828|
|[12995.0,0.0,0.0,...|14031|
|[13500.0,0.0,0.0,...|14500|
|[14309.0,27687.0,...|15000|
|[14950.0,9905.0,0...|14950|
|[14994.0,0.0,0.0,...|16975|
|[15303.0,38997.0,...|15599|
|[15395.0,0.0,0.0,...|16995|
|[15626.0,0.0,0.0,...|15726|
|[15919.0,0.0,1.0,...|19570|
+--------------------+-----+
only showing top 20 rows
```

Out[29]:

```
360992
```

In [30]:

```
train_df.count()
```

Out[30]:

```
1299720
```

In [31]:

```
test_df.count()
```

Out[31]:

```
360992
```

In [32]:

```
from pyspark.ml.regression import LinearRegression
```

In [33]:

```
lr = LinearRegression(featuresCol = 'Features', labelCol='msrp', maxIter=1000)
```

In [34]:
```
train_df.show(100)
```

```
+--------------------+-----+
|            Features| msrp|
+--------------------+-----+
|[12387.0,0.0,0.0,...|12387|
|[12970.0,0.0,0.0,...|12970|
|[15218.0,0.0,0.0,...|15218|
|[18755.0,0.0,0.0,...|18755|
|[36999.0,0.0,0.0,...|36999|
|[18276.0,0.0,1.0,...|18276|
|[22140.0,0.0,1.0,...|22140|
|[0.0,0.0,1.0,2.0,...|    0|
|[20494.0,0.0,0.0,...|20494|
|[19026.0,0.0,1.0,...|19026|
|[18951.0,0.0,1.0,...|18951|
|[16649.0,0.0,0.0,...|16649|
|[16671.0,0.0,0.0,...|16671|
|[16998.0,0.0,0.0,...|16998|
|[20294.0,0.0,0.0,...|20294|
|[45996.0,0.0,0.0,...|45996|
|[23537.0,0.0,1.0,...|23537|
|[12999.0,0.0,0.0,...|12999|
|[13398.0,0.0,0.0,...|13398|
|[13692.0,0.0,0.0,...|14121|
|[20997.0,0.0,0.0,...|20997|
|[12568.0,0.0,1.0,...|12568|
|[14680.0,0.0,1.0,...|15680|
|[16829.0,0.0,1.0,...|17829|
|[12568.0,0.0,1.0,...|12568|
|[12959.0,0.0,0.0,...|12959|
|[12998.0,0.0,0.0,...|12998|
|[19514.0,0.0,1.0,...|19514|
|[12930.0,0.0,0.0,...|12930|
|[12970.0,0.0,0.0,...|12994|
|[20978.0,0.0,0.0,...|20978|
|[23652.0,0.0,1.0,...|23652|
|[10999.0,0.0,0.0,...|10999|
|[12575.0,0.0,0.0,...|12575|
|[24457.0,0.0,0.0,...|24457|
|[12799.0,0.0,0.0,...|12799|
|[14529.0,0.0,0.0,...|14529|
|[15683.0,0.0,0.0,...|15683|
|[24726.0,0.0,0.0,...|24726|
|[12572.0,0.0,1.0,...|12572|
|[17999.0,0.0,0.0,...|17999|
|[45644.0,0.0,0.0,...|45644|
|[12568.0,0.0,1.0,...|12568|
|[18868.0,0.0,1.0,...|19868|
|[24286.0,0.0,1.0,...|24286|
|[12394.0,0.0,0.0,...|12394|
|[14520.0,0.0,0.0,...|14999|
|[24885.0,0.0,0.0,...|24885|
|[24998.0,0.0,0.0,...|24998|
|[11999.0,0.0,0.0,...|11999|
|[15976.0,0.0,1.0,...|16476|
|[13769.0,0.0,0.0,...|13819|
|[12902.0,0.0,0.0,...|12902|
|[17645.0,0.0,0.0,...|17695|
|[12357.0,0.0,1.0,...|12357|
|[14384.0,0.0,0.0,...|14384|
|[18781.0,0.0,1.0,...|18781|
|[23550.0,0.0,1.0,...|23550|
|[13295.0,0.0,0.0,...|13295|
|[18328.0,0.0,0.0,...|18328|
|[26060.0,0.0,0.0,...|26060|
|[12949.0,0.0,0.0,...|12999|
|[16925.0,0.0,1.0,...|16925|
|[15900.0,0.0,0.0,...|15986|
|[12899.0,0.0,0.0,...|12900|
|[17821.0,0.0,0.0,...|17821|
|[33841.0,0.0,1.0,...|33841|
|[33125.0,0.0,1.0,...|33125|
|[12998.0,0.0,0.0,...|12998|
|[14621.0,0.0,0.0,...|14667|
|[20383.0,0.0,1.0,...|20383|
|[29288.0,0.0,1.0,...|29288|
|[36163.0,0.0,1.0,...|36163|
|[16860.0,0.0,1.0,...|16860|
|[12994.0,0.0,0.0,...|12994|
|[12995.0,0.0,0.0,...|13509|
|[12281.0,0.0,0.0,...|12685|
|[16462.0,0.0,1.0,...|17462|
|[23481.0,0.0,1.0,...|23481|
|[15999.0,0.0,0.0,...|16475|
|[18787.0,0.0,1.0,...|23787|
|[16648.0,0.0,0.0,...|16648|
|[18787.0,0.0,1.0,...|18787|
|[12687.0,0.0,0.0,...|12951|
|[12998.0,0.0,0.0,...|13596|
|[18851.0,0.0,0.0,...|18875|
|[16860.0,0.0,1.0,...|21360|
|[18787.0,0.0,1.0,...|20037|
|[20663.0,0.0,1.0,...|24163|
|[12407.0,0.0,1.0,...|12407|
|[12568.0,0.0,1.0,...|12568|
|[16860.0,0.0,1.0,...|21360|
|[24999.0,0.0,0.0,...|24999|
|[13998.0,0.0,0.0,...|13998|
|[21689.0,0.0,0.0,...|21689|
|[14800.0,0.0,0.0,...|14800|
|[16349.0,0.0,0.0,...|16428|
|[18272.0,0.0,0.0,...|18273|
|[10339.0,0.0,0.0,...|10752|
|[21935.0,0.0,0.0,...|21935|
+--------------------+-----+
only showing top 100 rows
```

In [35]:
```
test_df.show()
```

```
+--------------------+-----+
|            Features| msrp|
+--------------------+-----+
|[0.0,0.0,1.0,2.0,...|    0|
|[0.0,0.0,1.0,2.0,...|    0|
|[0.0,0.0,1.0,2.0,...|    0|
|[0.0,0.0,1.0,2.0,...|    0|
|[0.0,0.0,1.0,2.0,...|    0|
|[0.0,3.0,1.0,2.0,...|    0|
|[0.0,9.0,1.0,2.0,...|    0|
|[0.0,12153.0,0.0,...|    0|
|[0.0,15144.0,0.0,...|    0|
|[11499.0,45603.0,...|12399|
|[12928.0,0.0,1.0,...|16828|
|[12995.0,0.0,0.0,...|14031|
|[13500.0,0.0,0.0,...|14500|
|[14309.0,27687.0,...|15000|
|[14950.0,9905.0,0...|14950|
|[14994.0,0.0,0.0,...|16975|
|[15303.0,38997.0,...|15599|
|[15395.0,0.0,0.0,...|16995|
|[15626.0,0.0,0.0,...|15726|
|[15919.0,0.0,1.0,...|19570|
+--------------------+-----+
only showing top 20 rows
```

In [36]:
```
lr_model = lr.fit(train_df)
```

In [37]:
```
y_pred = lr_model.transform(test_df)
```

In [38]:
```
y_pred.show()
```

```
+--------------------+-----+-------------------+
|            Features| msrp|         prediction|
+--------------------+-----+-------------------+
|[0.0,0.0,1.0,2.0,...|    0| 1125.0722357587292|
|[0.0,0.0,1.0,2.0,...|    0| 16652.786260337907|
|[0.0,0.0,1.0,2.0,...|    0|-1202.6032615360382|
|[0.0,0.0,1.0,2.0,...|    0|  9599.820605161342|
|[0.0,0.0,1.0,2.0,...|    0|-15991.774871472644|
|[0.0,3.0,1.0,2.0,...|    0|  -17861.4783095269|
|[0.0,9.0,1.0,2.0,...|    0| 18427.370204653864|
|[0.0,12153.0,0.0,...|    0| 13243.816229516342|
|[0.0,15144.0,0.0,...|    0| 13290.599226041759|
|[11499.0,45603.0,...|12399|  42487.73347571959|
|[12928.0,0.0,1.0,...|16828| 27972.138046448712|
|[12995.0,0.0,0.0,...|14031|  40274.64985817963|
|[13500.0,0.0,0.0,...|14500| 31408.758727997727|
|[14309.0,27687.0,...|15000|  42971.42268637994|
|[14950.0,9905.0,0...|14950| 21933.404842900036|
|[14994.0,0.0,0.0,...|16975| 43991.068482176444|
|[15303.0,38997.0,...|15599|  39046.84608497289|
|[15395.0,0.0,0.0,...|16995|  16904.26493510482|
|[15626.0,0.0,0.0,...|15726|  24146.79598416574|
|[15919.0,0.0,1.0,...|19570| 12697.472309073622|
+--------------------+-----+-------------------+
only showing top 20 rows
```

In [39]:
```
y_pred.describe().show()
```

```
+-------+-------------------+--------------------+
|summary|               msrp|          prediction|
+-------+-------------------+--------------------+
|  count|             360992|              360992|
|   mean| 2000863.2584406303|  446060.76193985925|
| stddev|6.499651648471686E7| 3.225782477478744E7|
|    min|                  0|  -72687.01466037614|
|    max|         2147483647|2.5674029905667214E9|
+-------+-------------------+--------------------+
```

In [40]:
```
print("Coefficients: " + str(lr_model.coefficients))
print("Intercept: " + str(lr_model.intercept))
```

```
Coefficients: [1.1955130707373247,0.39095921094982955,-4256.668333671143,-960.4479598289017,-4513.4733
53312087,4829.620689285545,0.0,-0.02818322086852489,-0.026681537555876916,-0.45795822902252364,-19.074
779173113654,3924.2135531015383,-2861.0278897925405,-54.87006199341723,36.44750887391078,-4903.3861310
88277,-2160.188891211929,290.9167649494918,-57.52588882733706,-0.5990587556657906,3441.0627458695367,-
2841.2497410185674,3197.880073594043,-5646.643740351718,84.31609582145248]
Intercept: -19468.11755917413
```

In [41]:
```
trainingSummary = lr_model.summary
print("RMSE: %f" % trainingSummary.rootMeanSquaredError)
print("r2: %f" % trainingSummary.r2)
```

```
RMSE: 3938430.054993
r2: 0.000040
```

In [42]:
```
y_pred.select("prediction","msrp","Features").show()
from pyspark.ml.evaluation import RegressionEvaluator
lr_evaluator = RegressionEvaluator(predictionCol="prediction", labelCol="msrp",metricName="r2")
print("R Squared (R2) on test data = %g" % lr_evaluator.evaluate(y_pred))
```

```
+--------------------+-----+--------------------+
|          prediction| msrp|            Features|
+--------------------+-----+--------------------+
| 1125.0722357587292|    0|[0.0,0.0,1.0,2.0,...|
| 16652.786260337907|    0|[0.0,0.0,1.0,2.0,...|
|-1202.6032615360382|    0|[0.0,0.0,1.0,2.0,...|
|  9599.820605161342|    0|[0.0,0.0,1.0,2.0,...|
|-15991.774871472644|    0|[0.0,0.0,1.0,2.0,...|
|  -17861.4783095269|    0|[0.0,3.0,1.0,2.0,...|
| 18427.370204653864|    0|[0.0,9.0,1.0,2.0,...|
| 13243.816229516342|    0|[0.0,12153.0,0.0,...|
| 13290.599226041759|    0|[0.0,15144.0,0.0,...|
|  42487.73347571959|12399|[11499.0,45603.0,...|
| 27972.138046448712|16828|[12928.0,0.0,1.0,...|
|  40274.64985817963|14031|[12995.0,0.0,0.0,...|
| 31408.758727997727|14500|[13500.0,0.0,0.0,...|
|  42971.42268637994|15000|[14309.0,27687.0,...|
| 21933.404842900036|14950|[14950.0,9905.0,0...|
| 43991.068482176444|16975|[14994.0,0.0,0.0,...|
|  39046.84608497289|15599|[15303.0,38997.0,...|
|  16904.26493510482|16995|[15395.0,0.0,0.0,...|
|  24146.79598416574|15726|[15626.0,0.0,0.0,...|
| 12697.472309073622|19570|[15919.0,0.0,1.0,...|
+--------------------+-----+--------------------+
only showing top 20 rows

R Squared (R2) on test data = 0.164857
```

In [43]:

```python
#Decision tree
from pyspark.ml.regression import DecisionTreeRegressor
dt = DecisionTreeRegressor(featuresCol ='Features', labelCol = 'msrp')
dt_model = dt.fit(train_df)
dt_pred = dt_model.transform(test_df)
dt_evaluator = RegressionEvaluator(
    labelCol="msrp", predictionCol="prediction", metricName="rmse")
rmse = dt_evaluator.evaluate(dt_pred)
dt_pred.select("prediction","msrp","Features").show(5)
print("Root Mean Squared Error (RMSE) on test data = %g" % rmse)
```

```
+----------+----+--------------------+
|prediction|msrp|            Features|
+----------+----+--------------------+
|       0.0|   0|[0.0,0.0,1.0,2.0,...|
|       0.0|   0|[0.0,0.0,1.0,2.0,...|
|       0.0|   0|[0.0,0.0,1.0,2.0,...|
|       0.0|   0|[0.0,0.0,1.0,2.0,...|
|       0.0|   0|[0.0,0.0,1.0,2.0,...|
+----------+----+--------------------+
only showing top 5 rows

Root Mean Squared Error (RMSE) on test data = 6.5025e+07
```

In [44]:

```python
#Gradient boosted tree regression

from pyspark.ml.regression import GBTRegressor
gbt = GBTRegressor(featuresCol = 'Features', labelCol = 'msrp', maxIter=10)
gbt_model = gbt.fit(train_df)
gbt_predictions = gbt_model.transform(test_df)
gbt_predictions.select('prediction', 'msrp', 'Features').show(5)
```

```
+-------------------+----+--------------------+
|         prediction|msrp|            Features|
+-------------------+----+--------------------+
| 1049.659896873842|   0|[0.0,0.0,1.0,2.0,...|
|-790.2891629690153|   0|[0.0,0.0,1.0,2.0,...|
|232.09748683109802|   0|[0.0,0.0,1.0,2.0,...|
| 1124.774128229476|   0|[0.0,0.0,1.0,2.0,...|
| 95.70418929729463|   0|[0.0,0.0,1.0,2.0,...|
+-------------------+----+--------------------+
only showing top 5 rows
```

In [45]:

```python
gbt_evaluator = RegressionEvaluator(
    labelCol="msrp", predictionCol="prediction", metricName="rmse")
rmse = gbt_evaluator.evaluate(gbt_predictions)
print("Root Mean Squared Error (RMSE) on test data = %g" % rmse)
```

```
Root Mean Squared Error (RMSE) on test data = 6.50247e+07
```

In [46]:

```
#Random Forest regression

from pyspark.ml.regression import RandomForestRegressor
rf = RandomForestRegressor(featuresCol = 'Features', labelCol = 'msrp')
rf_model = rf.fit(train_df)
rf_predictions = rf_model.transform(test_df)
rf_predictions.select('prediction', 'msrp', 'Features').show(5)
```

```
+------------------+----+--------------------+
|        prediction|msrp|            Features|
+------------------+----+--------------------+
|10450.204172665788|   0|[0.0,0.0,1.0,2.0,...|
|12339.358591085871|   0|[0.0,0.0,1.0,2.0,...|
|10912.716874344278|   0|[0.0,0.0,1.0,2.0,...|
| 17527.58963664177|   0|[0.0,0.0,1.0,2.0,...|
| 17342.74259213676|   0|[0.0,0.0,1.0,2.0,...|
+------------------+----+--------------------+
only showing top 5 rows
```

In [47]:

```
rf_evaluator = RegressionEvaluator(
    labelCol="msrp", predictionCol="prediction", metricName="rmse")
rmse = rf_evaluator.evaluate(rf_predictions)
print("Root Mean Squared Error (RMSE) on test data = %g" % rmse)
```

```
Root Mean Squared Error (RMSE) on test data = 6.50259e+07
```

In [48]:

```
#isotonic regressor

from pyspark.ml.regression import IsotonicRegression
ir = IsotonicRegression(featuresCol = 'Features', labelCol = 'msrp')
ir_model = ir.fit(train_df)
ir_predictions = ir_model.transform(test_df)
ir_predictions.select('prediction', 'msrp', 'Features').show(5)
```

```
+----------+----+--------------------+
|prediction|msrp|            Features|
+----------+----+--------------------+
|       0.0|   0|[0.0,0.0,1.0,2.0,...|
|       0.0|   0|[0.0,0.0,1.0,2.0,...|
|       0.0|   0|[0.0,0.0,1.0,2.0,...|
|       0.0|   0|[0.0,0.0,1.0,2.0,...|
|       0.0|   0|[0.0,0.0,1.0,2.0,...|
+----------+----+--------------------+
only showing top 5 rows
```

In [49]:

```
ir_evaluator = RegressionEvaluator(
    labelCol="msrp", predictionCol="prediction", metricName="rmse")
rmse = ir_evaluator.evaluate(ir_predictions)
print("Root Mean Squared Error (RMSE) on test data = %g" % rmse)
```

```
Root Mean Squared Error (RMSE) on test data = 6.49957e+07
```

Specific Question Proposed

In [50]:

```
#
df = df.select('firstSeen',
 'lastSeen',
 'msrp',
 'askPrice',
 'mileage',
 (df.isNew.cast(IntegerType())),         ## Converting Boolean to Numeric
 'color',
 'brandName',
 'modelName')
```

In [51]:

```
df.count()
```

Out[51]:

```
5695015
```

In [52]:
```
df.na.drop(how = "any")
```

Out[52]:
```
DataFrame[firstSeen: timestamp, lastSeen: timestamp, msrp: int, askPrice: int, mileage: int, isNew: in
t, color: string, brandName: string, modelName: string]
```

In [53]:
```
df_car_pyspark = df.sample(0.1)
```

In [54]:
```
df_car_pyspark.count()
```

Out[54]:
```
570977
```

In [55]:
```
df_car_pyspark.printSchema()
```

```
root
 |-- firstSeen: timestamp (nullable = true)
 |-- lastSeen: timestamp (nullable = true)
 |-- msrp: integer (nullable = true)
 |-- askPrice: integer (nullable = true)
 |-- mileage: integer (nullable = true)
 |-- isNew: integer (nullable = true)
 |-- color: string (nullable = true)
 |-- brandName: string (nullable = true)
 |-- modelName: string (nullable = true)
```

In [56]:
```
df_car_pyspark.describe().show()
```

```
+-------+-------------------+-------------------+-----------------+-------------------+---------------
-----+---------+------------------+
|summary|               msrp|           askPrice|          mileage|              isNew|
color|brandName|         modelName|
+-------+-------------------+-------------------+-----------------+-------------------+---------------
-----+---------+------------------+
|  count|             570977|             570977|           570977|             570977|              5
70977|   570861|            570371|
|   mean|   720776.410953506| 177120.53963119356|22006.17455869501|0.34448497925485616|  16413.8904109
58906|     null|1352.6327305489415|
| stddev|3.864848769443795E7|1.819671471510693E7|45503.31890962189|0.47520045645206654|    49950.70672
27175|     null| 835.4543171304593|
|    min|                  0|                  0|                0|                  0|
|    ACURA|        124 Spider|
|    max|         2147483647|         2147483647|          9024018|                  1|white silver me
ta...|   YAMAHA|        new Passat|
+-------+-------------------+-------------------+-----------------+-------------------+---------------
-----+---------+------------------+
```

In [57]:
```
#Q1 How long brfore the car is sold?

from pyspark.sql.functions import datediff,col,lit

df1 = df_car_pyspark.withColumn('Time_in_lot',datediff(df_car_pyspark['lastSeen'],df_car_pyspark['First
```

In [58]:
```
df1 = df1.select('Time_in_lot')
```

In [59]:
```
df1.dtypes
```

Out[59]:
```
[('Time_in_lot', 'int')]
```

In [60]:
```
df1.sort(df1.Time_in_lot.desc()).show(5)
```

```
+-----------+
|Time_in_lot|
+-----------+
|       1396|
|       1395|
|       1369|
|       1365|
|       1364|
+-----------+
only showing top 5 rows
```

In [61]:
```
df1.describe().show()
```

```
+-------+------------------+
|summary|       Time_in_lot|
+-------+------------------+
|  count|            570977|
|   mean| 67.92468172973692|
| stddev|106.31335378028105|
|    min|             -1032|
|    max|              1396|
+-------+------------------+
```

As seen above, a summary of the timeInLot variable (measured in days) shows that the minimum number of days that a car was in a dealer’s lot was zero, whereas the max was 1,409 days (~3.9 years).

In [62]:

```
#Q2 Which Cars are the Most Popular?

from pyspark.sql.functions import concat,col,lit,asc,desc
```

In [63]:

```
df2 = df_car_pyspark.select(concat(col("brandName"),lit(" "),col("modelName")).alias("Cars"))
```

In [64]:

```
df2.show(10)
```

```
+--------------------+
|                Cars|
+--------------------+
|MITSUBISHI Eclips...|
|       NISSAN Altima|
|       FORD Explorer|
|           FORD Edge|
|         FORD Fusion|
|         FORD Fusion|
|         FORD Fusion|
|         FORD Fiesta|
|         LINCOLN MKX|
|         FORD Fiesta|
+--------------------+
only showing top 10 rows
```

In [65]:

```
count = df2.groupBy('Cars').count()
```

In [66]:

```
sparkcount_df = count.orderBy(col("count").desc())
```

In [67]:

```
sparkcount_df.show(5)
```

```
+-------------------+-----+
|               Cars|count|
+-------------------+-----+
|         FORD F-150|17366|
|CHEVROLET Silverado|16400|
|  CHEVROLET Equinox|15499|
|        FORD Escape|11810|
|JEEP Grand Cherokee|10780|
+-------------------+-----+
only showing top 5 rows
```

In [68]:

```
import matplotlib.pyplot as plt
import pandas as pd

pandascount_df = sparkcount_df.toPandas()
```

In [69]:

```
pandascount_df.set_index('Cars',inplace =True)
```

In [70]:

```
pandascount_df = pandascount_df.head(5)
```

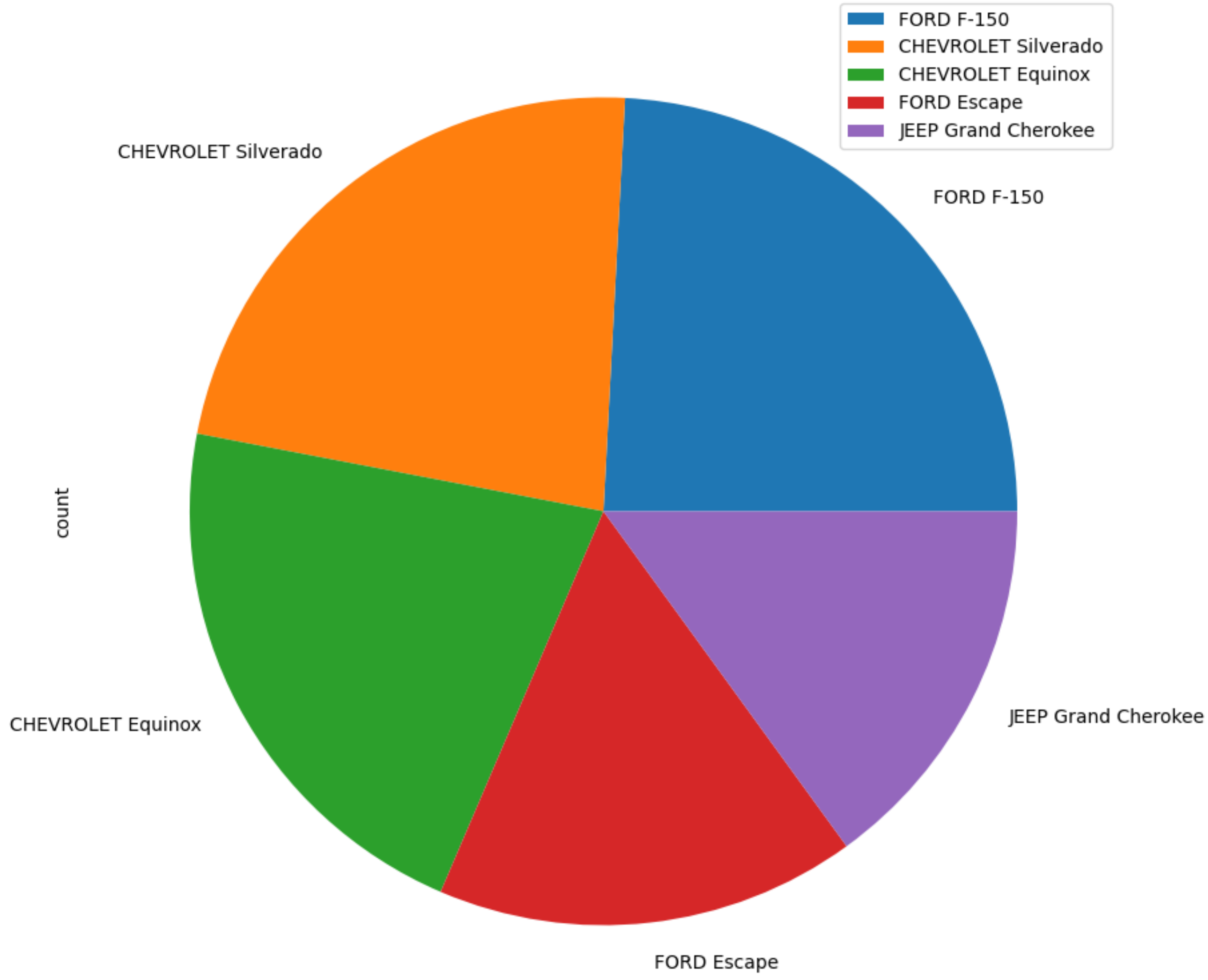
In [71]:

```
pandascount_df
```

Out[71]:

Cars	count
FORD F-150	17366
CHEVROLET Silverado	16400
CHEVROLET Equinox	15499
FORD Escape	11810
JEEP Grand Cherokee	10780

In [72]:
```
plot = pandascount_df.plot.pie(y= 'count', figsize = (10,15))
```



In [73]:
```
#Q3 What Color of the car do people like the most?

df3 = df_car_pyspark.select('color')
```

In [74]:
```
color = df3.groupby("color").count()
```

In [75]:
```
sparkcolor_df = color.orderBy(col("count").desc())
```

In [76]:
```
sparkcolor_df.show(6)
```

```
+------------+------+
|       color| count|
+------------+------+
|         N/A|211012|
|       Black| 21084|
|       White| 11049|
|        Gray|  7887|
|Summit White|  7221|
|      Silver|  6794|
+------------+------+
only showing top 6 rows
```

In [77]:

```
sparkcolor_df = sparkcolor_df.filter(sparkcolor_df.color!= 'N/A')
```

In [78]:

```
colorpandas_df = sparkcolor_df.toPandas()
```

In [79]:

```
colorpandas_df.set_index('color',inplace =True)
```

In [80]:

```
colorpandas_df = colorpandas_df.head(6)
```

In [81]:

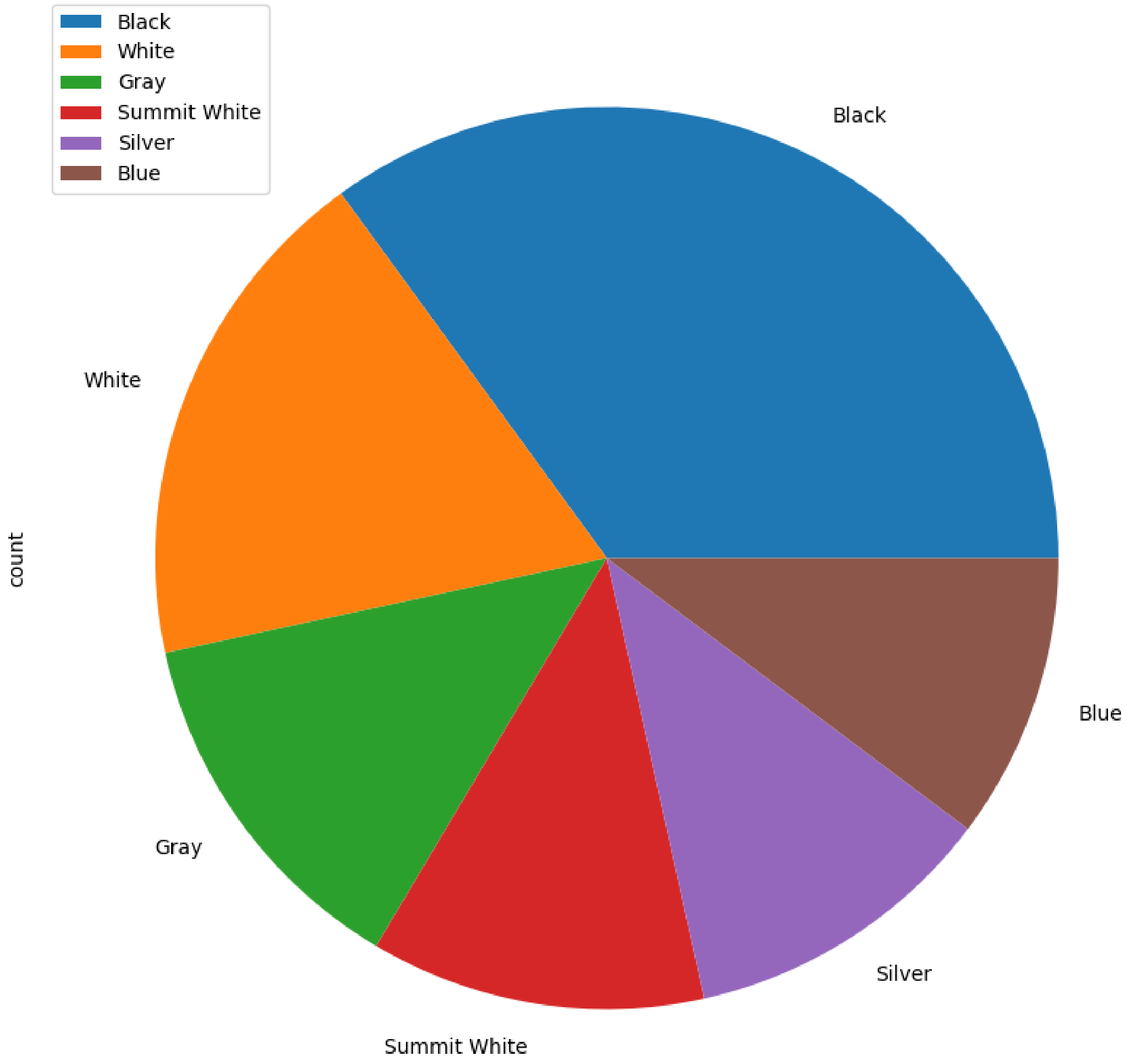
```
colorpandas_df
```

Out[81]:

	count
color	
Black	21084
White	11049
Gray	7887
Summit White	7221
Silver	6794
Blue	6171

In [82]:

```
plot = colorpandas_df.plot.pie(y= 'count', figsize = (10,15))
```



```
In [83]: #Q5 How much will the final price differ from MSRP on avg

         df5 = df_car_pyspark.select('msrp','askPrice')
```

```
In [84]: df5.printSchema()
```

```
root
 |-- msrp: integer (nullable = true)
 |-- askPrice: integer (nullable = true)
```

```
In [85]: df5.show()
```

```
+-----+--------+
| msrp|askPrice|
+-----+--------+
| 1498|    1498|
|10589|   10589|
|36999|   36999|
|23652|   23652|
|12575|   12575|
|15683|   15683|
|11763|   11763|
|12572|   12572|
|24885|   24885|
| 7685|    7650|
|18781|   18781|
|26060|   26060|
|36163|   36163|
|13509|   12995|
|12685|   12281|
|17321|   16993|
|20037|   18787|
|23888|   22396|
|12689|   12689|
|    0|       0|
+-----+--------+
only showing top 20 rows
```

In [86]:

```
df5.corr("msrp","askPrice")
```

Out[86]:

```
0.4707036907884924
```

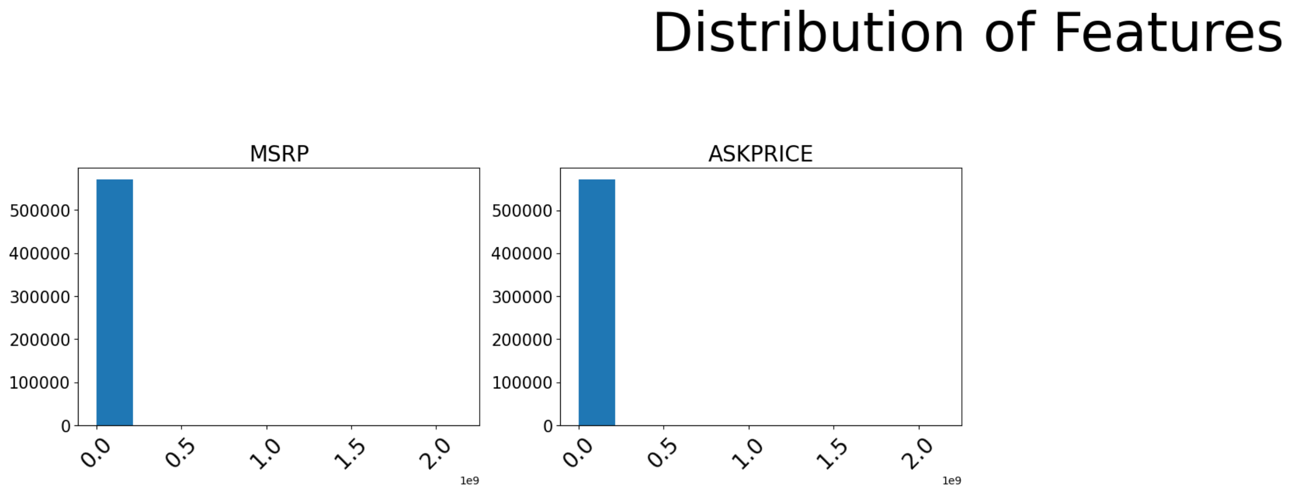
In [87]:

```
import matplotlib.pyplot as plt
import seaborn as sns
```

In [88]:

```
fig = plt.figure(figsize=(25, 15))
st = fig.suptitle("Distribution of Features", fontsize=50, verticalalignment="baseline")
for col, num in zip(df5.toPandas().describe().columns, range(1,11)):
  ax = fig.add_subplot(3,4, num)
  ax.hist(df5.toPandas()[col])
  plt.grid(False)
  plt.xticks(rotation=45, fontsize=20)
  plt.yticks(fontsize=15)
  plt.title(col.upper(), fontsize=20)

plt.tight_layout()
st.set_y(0.95)
fig.subplots_adjust(top=0.85, hspace=0.4)
plt.show()
```



In [89]:

```
df5 = df5.withColumn('Price_diff', df5['msrp']- df5['askPrice'])
```

In [90]:

```
df5.show()
```

```
+-----+--------+----------+
| msrp|askPrice|Price_diff|
+-----+--------+----------+
| 1498|    1498|         0|
|10589|   10589|         0|
|36999|   36999|         0|
|23652|   23652|         0|
|12575|   12575|         0|
|15683|   15683|         0|
|11763|   11763|         0|
|12572|   12572|         0|
|24885|   24885|         0|
| 7685|    7650|        35|
|18781|   18781|         0|
|26060|   26060|         0|
|36163|   36163|         0|
|13509|   12995|       514|
|12685|   12281|       404|
|17321|   16993|       328|
|20037|   18787|      1250|
|23888|   22396|      1492|
|12689|   12689|         0|
|    0|       0|         0|
+-----+--------+----------+
only showing top 20 rows
```

In [91]:
```
df5.select('price_diff').describe().show()
```

```
+-------+-------------------+
|summary|         price_diff|
+-------+-------------------+
|  count|             570977|
|   mean|  543655.8713223125|
| stddev|3.409922691264347E7|
|    min|                  0|
|    max|         2147480657|
+-------+-------------------+
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: