Explode function in pyspark

Explode() function in pyspark

pyspark adopts parallel nomenclature with pandas package in "explode" function

explode function exclusively applys on MapType or ArrayType, and unnest applied row contents in usage

There are four related function of "explode"

1. explode
2. explode_outer
3. posexplode
4. posexplode_outer

Each funciont will be explained below

In [34]:
```
# prepare dataframes
from pyspark.sql.types import *
from pyspark.sql.types import *
df = spark.createDataFrame(
    [
        ('Eric',['swim','jogging'],{'car':['toyota','lexus']}),
        ('Amy',['jogging'],{'car':['BMW'],'scooter':['SYM']}),
        ('Alex',[],{}),
    ],
    StructType([StructField("name", StringType(), True),
                StructField("interest", ArrayType(StringType()), True),
                StructField("car_property", MapType(StringType(),ArrayType(StringTy
)
```

```
StatementMeta(sparkcluster01, 9809, 35, Finished, Available)
```

In [35]:
```
from pyspark.sql.functions import explode , explode_outer, posexplode, posexplode_o
display(df)
```

```
StatementMeta(sparkcluster01, 9809, 36, Finished, Available)
SynapseWidget(Synapse.DataFrame, d6b681ec-b64e-4f3e-86be-383b9f59f7e0)
```

explode function can only use in single 'select' clause

- Apply on ArrayType: The value in single rows will split into different rows after "explode"
- Apply on MapType: The key values pair in single rows will split into different columns after "explode"

In [36]:
```
display(
    df.select('name', explode(df.interest))

)
```

```
StatementMeta(sparkcluster01, 9809, 37, Finished, Available)
SynapseWidget(Synapse.DataFrame, 34c11031-69f6-4120-a9a3-eeb6ba1748b0)
```

In [37]:
```
display(
    df.select('name', explode(df.car_property))

)
```

StatementMeta(sparkcluster01, 9809, 38, Finished, Available)
SynapseWidget(Synapse.DataFrame, 5700c0b2-6858-4b7a-b3f8-eab19ffc71fe)

explode_outer function can keep null value which exist in the specific row

In [38]:
```
display(
    df.select('name', explode_outer(df.interest))

)
```

StatementMeta(sparkcluster01, 9809, 39, Finished, Available)
SynapseWidget(Synapse.DataFrame, acb52d4c-dcd3-4fcb-b7a6-15c3ccdc2915)

In [39]:
```
display(
    df.select('name', explode_outer(df.car_property))

)
```

StatementMeta(sparkcluster01, 9809, 40, Finished, Available)
SynapseWidget(Synapse.DataFrame, 6d25bbc6-3e72-4f4d-9a75-532aa6c9af2e)

Additionally, compare to simple explode function, "posexplode_outer" function create one another row holding the origin position of the arraytype/maptype element to new rows in column

In [43]:
```
display(
    df.select('name', posexplode(df.interest))

)
```

StatementMeta(sparkcluster01, 9809, 44, Finished, Available)
SynapseWidget(Synapse.DataFrame, 88648a9f-2c13-4be7-b86c-8a454acd7974)

In [42]:
```
display(
    df.select('name', posexplode(df.car_property))

)
```

StatementMeta(sparkcluster01, 9809, 43, Finished, Available)
SynapseWidget(Synapse.DataFrame, e33b4518-6f2f-4f46-9b18-8f04aa8919bf)

posexplode_outer function can keep null value which exist in the specific row similar to explode_outer

In [44]:
```
display(
    df.select('name', posexplode_outer(df.interest))
```

```
)
```

```
StatementMeta(sparkcluster01, 9809, 45, Finished, Available)
SynapseWidget(Synapse.DataFrame, e3f52ada-92eb-448b-9084-60d14a8ad483)
```

In [45]:
```
display(
    df.select('name', posexplode_outer(df.car_property))

)
```

```
StatementMeta(sparkcluster01, 9809, 46, Finished, Available)
SynapseWidget(Synapse.DataFrame, b9cbe48c-56e1-41bf-b288-63f05d024ce7)
```

In [ ]:
```

```