```
In [1]: import pyspark.pandas as ps
        from pyspark.sql import SparkSession
        import pandas as pd
        import numpy as np
        import warnings
        warnings.filterwarnings('ignore')
```

```
WARNING:root:'PYARROW_IGNORE_TIMEZONE' environment variable was not set. It is required to set this environment varia
ble to '1' in both driver and executor sides if you use pyarrow>=2.0.0. pandas-on-Spark will set it for you but it do
es not work if there is a Spark context already launched.
```

Create Pandas on Spark DataFrame

In [2]:
```
ps_df=ps.DataFrame([['France','50M','3T'],['India','30M','30T'],['Kenya','70M','25T'],
                     ['Nigeria','90M','60T'],['China','20M','2T'],['USA','80M','30T'],
                     ['UK','70M','25T'],['USA','20M','30T'],['China','70M','25T'],
                     ['France', '50M', '3T'],['China','70M','25T'] ],
                          columns=['Country','Population','GDP'])
ps_df
```

```
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
22/01/14 11:47:59 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java
classes where applicable
```

Out[2]:

	Country	Population	GDP
0	France	50M	3T
1	India	30M	30T
2	Kenya	70M	25T
3	Nigeria	90M	60T
4	China	20M	2T
5	USA	80M	30T
6	UK	70M	25T
7	USA	20M	30T
8	China	70M	25T
9	France	50M	3T
10	China	70M	25T

```
In [3]: salary_df = ps.DataFrame(
            {
                "Department": ["Finance","Technology","Finance","Technology","Technology"],
                "Staff": ["Tom", "Peter","Simon", "Mary", "Jane"],
                "Salary": [90000.00, 57000.00,40000.00, 34000.00, 12000.00]
            },
        )

        salary_df
```

Out[3]:

	Department	Staff	Salary
0	Finance	Tom	90000.0
1	Technology	Peter	57000.0
2	Finance	Simon	40000.0
3	Technology	Mary	34000.0
4	Technology	Jane	12000.0

Create Spark DataFrame

```
In [4]: spark = SparkSession.builder.getOrCreate()
        sdf=spark.createDataFrame([['France','50M','3T'],['India','30M','30T'],['Kenya','70M','25T'],
                                   ['Nigeria','90M','60T'],['China','20M','2T'],['USA','80M','30T'],
                                   ['UK','70M','25T'],['USA','20M','30T'],['China','70M','25T'],
                                   ['France', '50M', '3T'],['China','70M','25T'] ],
                                        schema='Country string,Population string,GDP string')
        sdf.show()
```

```
+-------+----------+---+
|Country|Population|GDP|
+-------+----------+---+
| France|       50M| 3T|
|  India|       30M|30T|
|  Kenya|       70M|25T|
|Nigeria|       90M|60T|
|  China|       20M| 2T|
|    USA|       80M|30T|
|     UK|       70M|25T|
|    USA|       20M|30T|
|  China|       70M|25T|
| France|       50M| 3T|
|  China|       70M|25T|
+-------+----------+---+
```

Create Pandas DataFrame

In [5]:
```
pd_df=pd.DataFrame([['France','50M','3T'],['India','30M','30T'],['Kenya','70M','25T'],
                     ['Nigeria','90M','60T'],['China','20M','2T'],['USA','80M','30T'],
                     ['UK','70M','25T'],['USA','20M','30T'],['China','70M','25T'],
                     ['France', '50M', '3T'],['China','70M','25T'] ],
                          columns=['Country','Population','GDP'])
pd_df
```

Out[5]:

	Country	**Population**	**GDP**
0	France	50M	3T
1	India	30M	30T
2	Kenya	70M	25T
3	Nigeria	90M	60T
4	China	20M	2T
5	USA	80M	30T
6	UK	70M	25T
7	USA	20M	30T
8	China	70M	25T
9	France	50M	3T
10	China	70M	25T

Read external csv file with Pandas-on-SPark

In [6]:
```
df=ps.read_csv("titanic.csv")
```

In [7]:
```
df.head()
```

Read csv data with Pandas-on-Saprk

```
In [8]: ps_to_pd_df=ps_df.to_pandas()
        ps_to_pd_df
```

```
Out[8]:
```

	Country	Population	GDP
0	France	50M	3T
1	India	30M	30T
2	Kenya	70M	25T
3	Nigeria	90M	60T
4	China	20M	2T
5	USA	80M	30T
6	UK	70M	25T
7	USA	20M	30T
8	China	70M	25T
9	France	50M	3T
10	China	70M	25T

Convert Pandas DataFrame to Pandas-on-Spark DataFrame

In [9]:
```
pd_to_ps_df=ps.from_pandas(pd_df)
pd_to_ps_df
```

Out[9]:

	Country	Population	GDP
0	France	50M	3T
1	India	30M	30T
2	Kenya	70M	25T
3	Nigeria	90M	60T
4	China	20M	2T
5	USA	80M	30T
6	UK	70M	25T
7	USA	20M	30T
8	China	70M	25T
9	France	50M	3T
10	China	70M	25T

Convert Pandas DataFrame to Spark DataFrame

```
In [10]: pd_to_sdf=spark.createDataFrame(pd_df)
         pd_to_sdf.show()
```

```
+-------+----------+---+
|Country|Population|GDP|
+-------+----------+---+
| France|       50M| 3T|
|  India|       30M|30T|
|  Kenya|       70M|25T|
|Nigeria|       90M|60T|
|  China|       20M| 2T|
|    USA|       80M|30T|
|     UK|       70M|25T|
|    USA|       20M|30T|
|  China|       70M|25T|
| France|       50M| 3T|
|  China|       70M|25T|
+-------+----------+---+
```

Pandas on Spark Functions

```
In [11]: salary_df
```

Out[11]:

	Department	Staff	Salary
0	Finance	Tom	90000.0
1	Technology	Peter	57000.0
2	Finance	Simon	40000.0
3	Technology	Mary	34000.0
4	Technology	Jane	12000.0

Check rows and columns

```
In [12]: salary_df.shape
```

```
Out[12]: (5, 3)
```

Check DataFrame types

```
In [13]: salary_df.info()
```

```
<class 'pyspark.pandas.frame.DataFrame'>
Int64Index: 5 entries, 0 to 4
Data columns (total 3 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   Department  5 non-null      object
 1   Staff       5 non-null      object
 2   Salary      5 non-null      float64
dtypes: float64(1), object(2)
```

Check Statistical Summary

```
In [14]: salary_df.describe()
```

```
Out[14]:
```

	Salary
count	5.000000
mean	46600.000000
std	29117.005341
min	12000.000000
25%	34000.000000
50%	40000.000000
75%	57000.000000
max	90000.000000

Calculate Sum

```
In [15]: salary_df['Salary'].sum()
```

```
Out[15]: 233000.0
```

Calculate Mean

```
In [16]: salary_df['Salary'].mean()
```

```
Out[16]: 46600.0
```

Calculate Standard Deviation

```
In [17]: salary_df['Salary'].std()

Out[17]: 29117.005340522228
```

Calculate Variance of Salary

```
In [18]: salary_df['Salary'].var()

Out[18]: 847800000.0
```

Calculate Skewnes of Salary

```
In [19]: salary_df['Salary'].skew()

Out[19]: 0.44342185901218767
```

Group Salary by Department

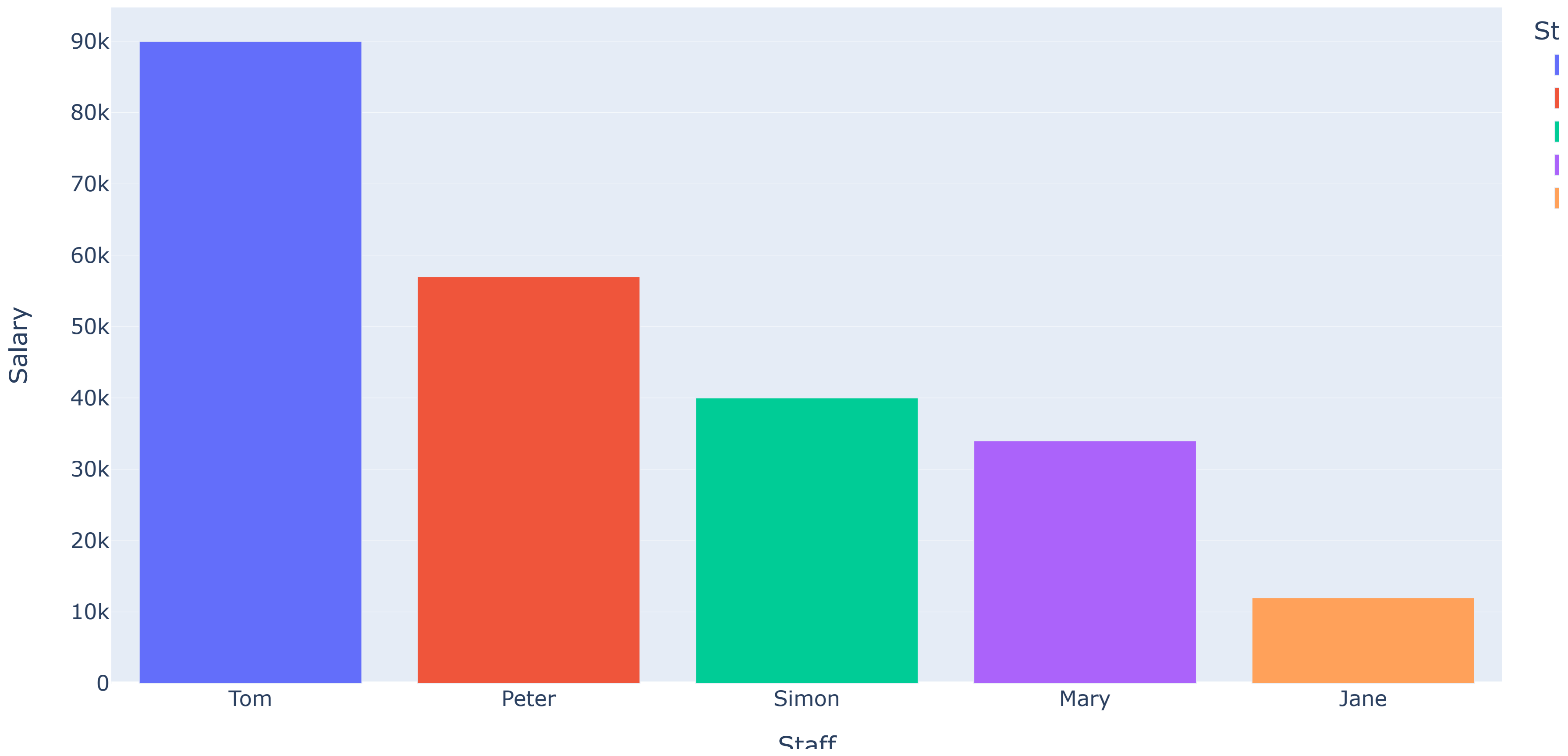
```
In [20]: salary_df.groupby('Department')['Salary'].sum()

Out[20]: Department
         Finance       130000.0
         Technology    103000.0
         Name: Salary, dtype: float64
```

Ploting Visualizations in Pandas on Spark

Pandas on Spark leverages plotly in the backend for visualization

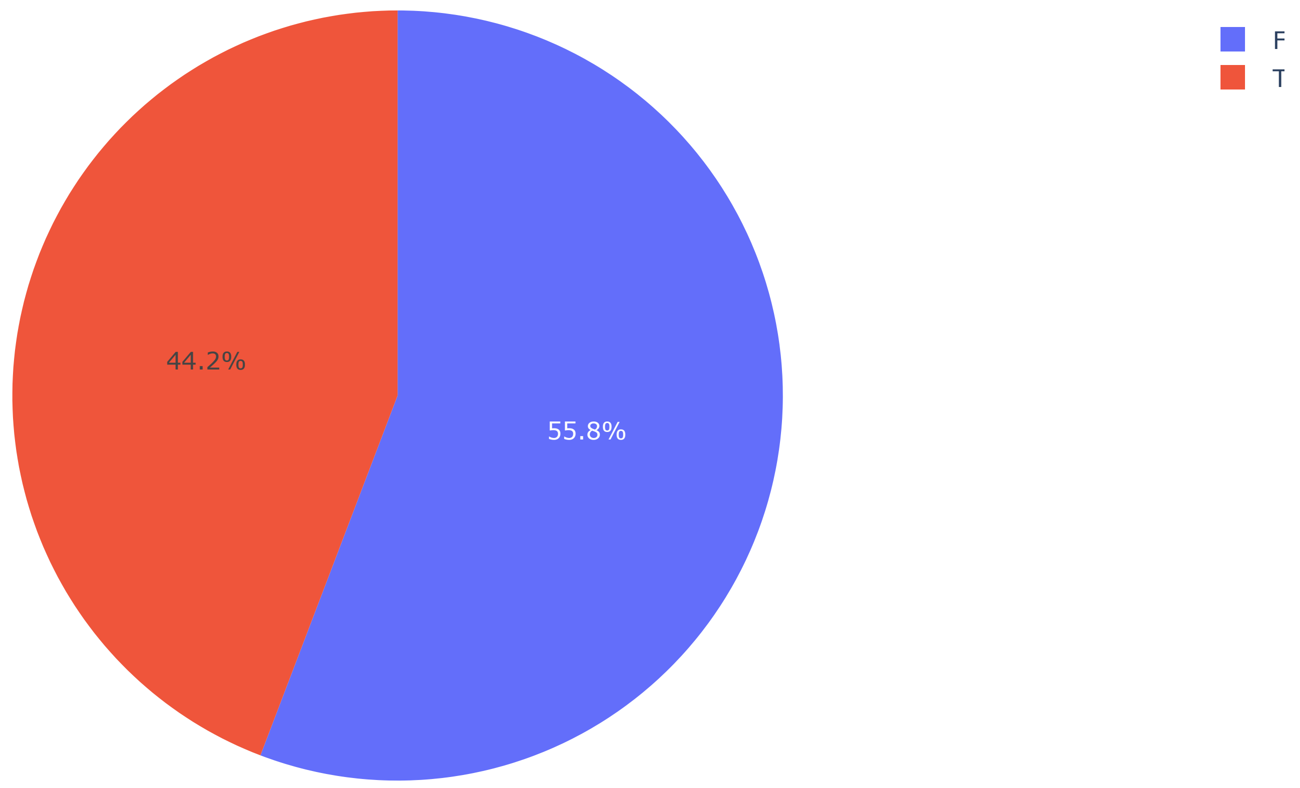
Let's plot Salary for each Staff on a bar graph

```
In [21]: salary_df.plot.bar(x='Staff',y='Salary',color='Staff')
```



Pie Chart of the Salary per Department

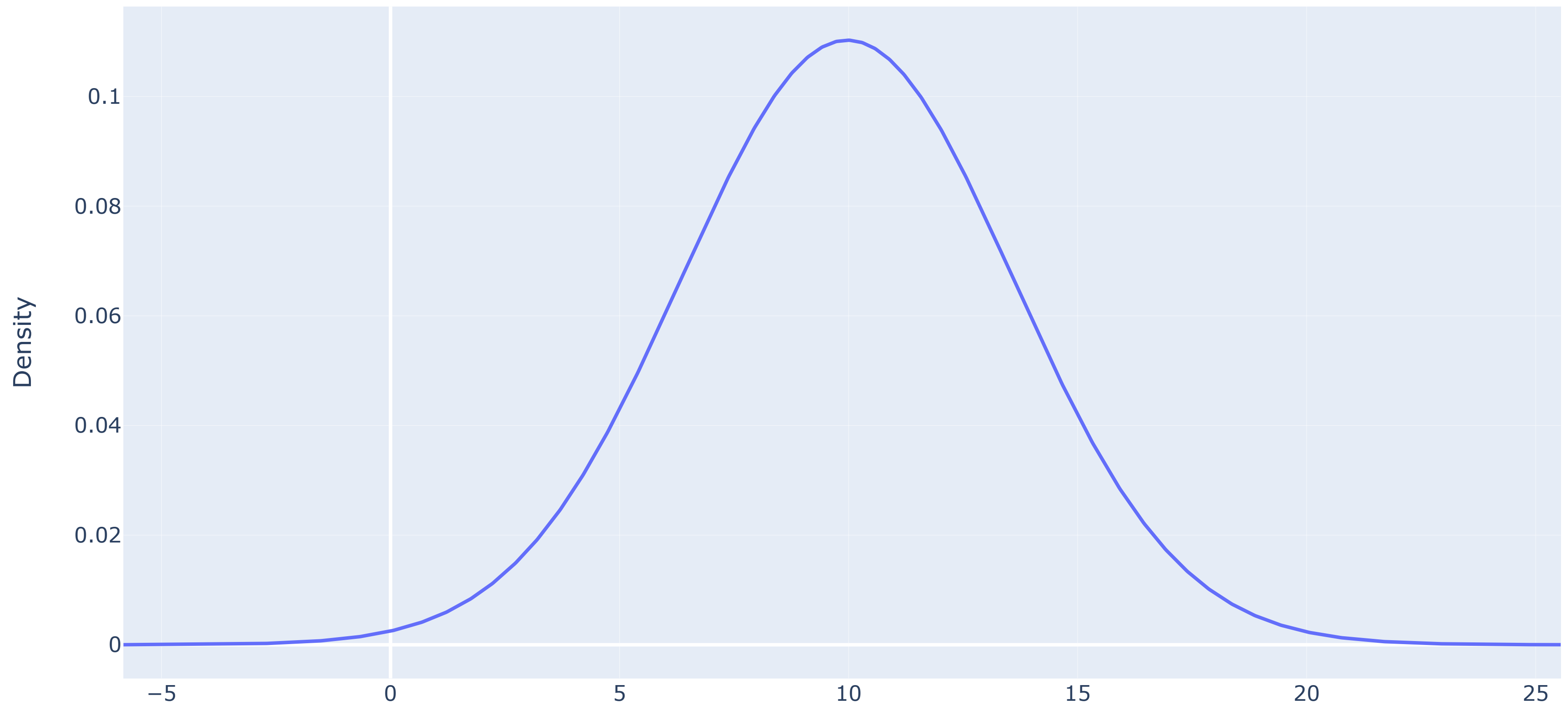
In [22]:
```
salary_df.groupby('Department')['Salary'].sum().plot.pie()
```



Kernel Density Estimation for a normal distribution data

```
In [23]: ps.DataFrame(np.random.normal(10,2,10000)).plot.kde(bw_method=3)
```

```
22/01/14 11:48:38 WARN InstanceBuilder$NativeBLAS: Failed to load implementation from:dev.ludovic.netlib.blas.JNIBLAS
22/01/14 11:48:39 WARN InstanceBuilder$NativeBLAS: Failed to load implementation from:dev.ludovic.netlib.blas.Foreign
LinkerBLAS
```


Density