```
import org.apache.spark._
import org.apache.spark.rdd.RDD
import org.apache.spark.graphx._
```

```
scala> import org.apache.spark._
import org.apache.spark._

scala> import org.apache.spark.rdd.RDD
import org.apache.spark.rdd.RDD

scala> import org.apache.spark.graphx._
import org.apache.spark.graphx._
```

```
val vertices = Array((1L,("A")),(2L,("B")),(3L,("C")))
```

```
scala> val vertices = Array((1L,("A")),(2L,("B")),(3L,("C")))
val vertices: Array[(Long, String)] = Array((1,A), (2,B), (3,C))
```

```
val vRDD = sc.parallelize(vertices)
```

```
scala> val vRDD = sc.parallelize(vertices)
warning: 1 deprecation (since 2.13.0); for details, enable `:setting -deprecation` or `:replay -deprec
ation`
val vRDD: org.apache.spark.rdd.RDD[(Long, String)] = ParallelCollectionRDD[0] at parallelize at <conso
le>:1
```

```
vRDD.take(1)
vRDD.take(2)
```

```
scala> vRDD.take(1)
val res0: Array[(Long, String)] = Array((1,A))

scala> vRDD.take(2)
val res1: Array[(Long, String)] = Array((1,A), (2,B))
```

val edges = Array(Edge(1L,2L,1800),Edge(2L,3L,800),Edge(3L,1L,1400))

```
scala> val edges = Array(Edge(1L,2L,1800),Edge(2L,3L,800),Edge(3L,1L,1400))
val edges: Array[org.apache.spark.graphx.Edge[Int]] = Array(Edge(1,2,1800), Edge(2,3,800), Edge(3,1,14
00))
```

val eRDD = sc.parallelize(edges)

```
scala> val eRDD = sc.parallelize(edges)
warning: 1 deprecation (since 2.13.0); for details, enable `:setting -deprecation` or `:replay -deprec
ation`
val eRDD: org.apache.spark.rdd.RDD[org.apache.spark.graphx.Edge[Int]] = ParallelCollectionRDD[1] at pa
rallelize at <console>:1
```

eRDD.take(2)

```
scala> eRDD.take(2)
val res2: Array[org.apache.spark.graphx.Edge[Int]] = Array(Edge(1,2,1800), Edge(2,3,800))
```

val nowhere = "nowhere"

```
scala> val nowhere = "nowhere"
val nowhere: String = nowhere
```

val graph = Graph(vRDD,eRDD,nowhere)

```
scala> val graph = Graph(vRDD,eRDD,nowhere)
val graph: org.apache.spark.graphx.Graph[String,Int] = org.apache.spark.graphx.impl.GraphImpl@2e008502
```

#To check number of Airports

val numairports = graph.numVertices

```
scala> val numairports = graph.numVertices
val numairports: Long = 3
```

#To check routes

val numairports = graph.numEdges

```
scala> val numairports = graph.numEdges
val numairports: Long = 3
```

#Route having distance > 1000

(graph.edges.filter{case Edge(src,dst,prop)=>prop>1000}.collect.foreach(println))

```
scala> (graph.edges.filter{case Edge(src,dst,prop)=>prop>1000}.collect.foreach(println))
warning: 1 deprecation (since 2.13.3); for details, enable `:setting -deprecation` or `:replay -deprec
ation`
Edge(1,2,1800)
Edge(3,1,1400)
```

#Triplet Information

graph.triplets.take(3).foreach(println)

```
scala> graph.triplets.take(3).foreach(println)
((1,A),(2,B),1800)
((2,B),(3,C),800)
((3,C),(1,A),1400)
```

#Indegree

val i = graph.inDegrees

i.collect()

```
scala> val i = graph.inDegrees
val i: org.apache.spark.graphx.VertexRDD[Int] = VertexRDDImpl[25] at RDD at VertexRDD.scala:57

scala> i.collect()
val res5: Array[(org.apache.spark.graphx.VertexId, Int)] = Array((1,1), (2,1), (3,1))
```

#Outdegrees

val o = graph.outDegrees

o.collect()

```
scala> val o = graph.outDegrees
val o: org.apache.spark.graphx.VertexRDD[Int] = VertexRDDImpl[29] at RDD at VertexRDD.scala:57

scala> o.collect()
val res6: Array[(org.apache.spark.graphx.VertexId, Int)] = Array((1,1), (2,1), (3,1))
```

#Total Degree

val t = graph.degrees

t.collect()

```
scala> val t = graph.degrees
val t: org.apache.spark.graphx.VertexRDD[Int] = VertexRDDImpl[33] at RDD at VertexRDD.scala:57

scala> t.collect()
val res8: Array[(org.apache.spark.graphx.VertexId, Int)] = Array((1,2), (2,2), (3,2))
```