**def** ParallelSort (InputTable, SortingColumnName, OutputTable, openconnection):  
 *#Implement ParallelSort Here.  
 #before\_test\_script\_starts\_middleware(openconnection,DATABASE\_NAME)* cur= openconnection.cursor()  
 query=**"""Select MIN("""**+str(SortingColumnName)+**""") from """**+str(InputTable)  
 cur.execute(query)  
 min=cur.fetchone()[0]  
 query=**"""Select MAX("""**+str(SortingColumnName)+**""") from """**+str(InputTable)  
 cur.execute(query)  
 max=cur.fetchone()[0]  
 equipartition(InputTable,5,SortingColumnName,min,max,openconnection)  
 *# cur = openconnection.cursor()* query=**"""SELECT \* INTO """**+str(OutputTable)+**""" FROM """**+str(InputTable)+**""" WHERE 1 = 0"""** cur.execute(query)  
 thread\_list=[]  
 delay=[0,2,4,6,8]  
 **for** i **in** range(5):  
 t = threading.Thread(target=InsertToTable, args=(**'Thread'**+str(i),InputTable,openconnection,i,delay[i],OutputTable,SortingColumnName))  
 t.start()  
 thread\_list.append(t)  
 **for** t **in** thread\_list:  
 t.join()  
 **pass**

**def** InsertToTable( threadName,InputTable,openconnection,num,delay,OutputTable,SortingColumnName):  
 cur=openconnection.cursor()  
 time.sleep(delay)  
 query=**""" Select \* from range\_"""**+str(InputTable)+str(num)+**""" order by """**+str(SortingColumnName)  
 cur.execute(query)  
 tuples=cur.fetchall()  
 insert=**""  
 for** each **in** tuples:  
 insert+=**"("  
 for** number **in** each:  
 **if**(isinstance(number,str)):  
 insert+=**"'"**+number+**"',"  
 else**:  
 insert+=str(number)+**","** insert =insert[:-1]  
 insert+=**"),"** insert=insert[:-1]  
 insertion=**"Insert into "**+str(OutputTable)+**" values "**+insert  
  
 **if**(threadName==**'Thread0'**):  
 cur.execute(insertion)  
 **print** threadName  
 semaphores[0].release()  
 **elif**(threadName==**'Thread1'**):  
 semaphores[0].acquire()  
 **print** threadName  
 cur.execute(insertion)  
 semaphores[1].release()  
 **elif**(threadName==**'Thread2'**):  
 semaphores[1].acquire()  
 **print** threadName  
 cur.execute(insertion)  
 semaphores[2].release()  
 **elif**(threadName==**'Thread3'**):  
 semaphores[2].acquire()  
 **print** threadName  
 cur.execute(insertion)  
 semaphores[3].release()  
 **elif**(threadName==**'Thread4'**):  
 semaphores[3].acquire()  
 **print** threadName  
 cur.execute(insertion)  
  
 **return**

**Explanation:** In the function parallelSort() I am horizontally dividing the InputTable into 5 partitions. Then I am creating 5 threads where each thread is responsible for sorting each partition which is done thread function InputToTable(). But my main task is to parallel run all 5 threads and do sorting of each partition table associated with the thread and store the data in the output table.

As I am dealing with threads, program doesn’t have the control over the threads. And threads execute based on the OS scheduling Algorithm. So Synchronization among the threads is main thing. I agree that each threads does the sorting of associated partition table but it would miss the pattern that it should store into output table( as I am intended to have sorted output table).

So in order to have synchronization among the threads I have used **semaphores.**