Code

```
    # Running Algorithms on the same student Data:
same_student_times = {}

test_same_student_data_db = studentDb()
test_same_student_data_db.make_students(1) # creating a db with 1
student
test_same_student_data_db.students =
[test_same_student_data_db.students[0] for i in range(20)] # 20 of
same student
test_same_student_data_db.write_to_db() # overwriting db with 20
copies of same student

print(test_same_student_data_db.students)

for p in params:
    # run and time each algo over the file of same students
    same_student_times[p] = {f: get_run_times(sort_funcs[f],
param=p) for f in sort_funcs.keys()}
    print(f"\n{p}: {same_student_times[p]}")
```

in the code above I first create a new studentDB() and populate it with a single student. I then set the students attribute of the DB to hold a list of 20 copies of the same student. I then run each of the four sorting algorithms over this database for parameters id, f_name, l_name.

Output

Here I am showing that the same student data is repeated in the DB file.

```
[{'id': 7599746813, 'f_name': 'nia', 'l_name': 'wong', 'email':
'niw8912@psu.edu', 'major': 'graphic design'}, {'id': 7599746813,
'f_name': 'nia', 'l_name': 'wong', ...
```

Below are the sort time for each parameter using the four different sorting algorithms.

```
id: {'insertion': 6.100000000008876e-08, 'selection':
5.0000000000050006e-08, 'bubble': 4.900000000002125e-08, 'merge':
8.500000000022378e-08}

f_name: {'insertion': 4.999999999977245e-08, 'selection':
1.940000000000275e-07, 'bubble': 1.869999999998262e-07, 'merge':
7.699999999999374e-08}

l_name: {'insertion': 5.0000000000050006e-08, 'selection':
4.699999999996374e-08, 'bubble': 4.600000000021254e-08, 'merge':
7.900000000005125e-08}

Process finished with exit code 0
```

Results

below is a table showing the CPU times for sorting the same student data:

Category	Bubble	Selection	Insertion	Merge
id	4.90e-08	5.00e-08	6.10e-08	8.50e-08
f_name	1.87e-07	1.94e-07	4.99e-08	7.70e-08
l_name	4.60e-08	4.70e-08	5.00e-08	7.90e-08

Fastest Times

- id: bubble
- f_name: bubble

- l_name: bubble

Conclusion

for the same student data, it makes sense that bubble sort would perform much quicker than the other sorting algorithms. This is because the while loop in bubble sort essentially is never entered and is basically turned into a conditional check. For the case of sorted student data, bubble sort has time complexity of $O(n)$.