

Python test results for each Task. Refer to python script or SQL queries in the SQL folder to see SQL queries for each task.

For original tables, refer to .sql test data file in ../SQL files/Tables-Test_data-DDL.sql

PYTHON TESTS:

TAKS A)

TEST 1 PASS A 100,A. N.,ano@somewhere.net

The screenshot shows a Python script running in a terminal. The script processes input data and outputs results. The input data is shown in a table on the right, and the output is shown in a table on the left. The script is written in Python and uses a database connection to insert and retrieve data.

OUTPUT

sno	sname	semail
100	A. N.	ano@somewhere.net

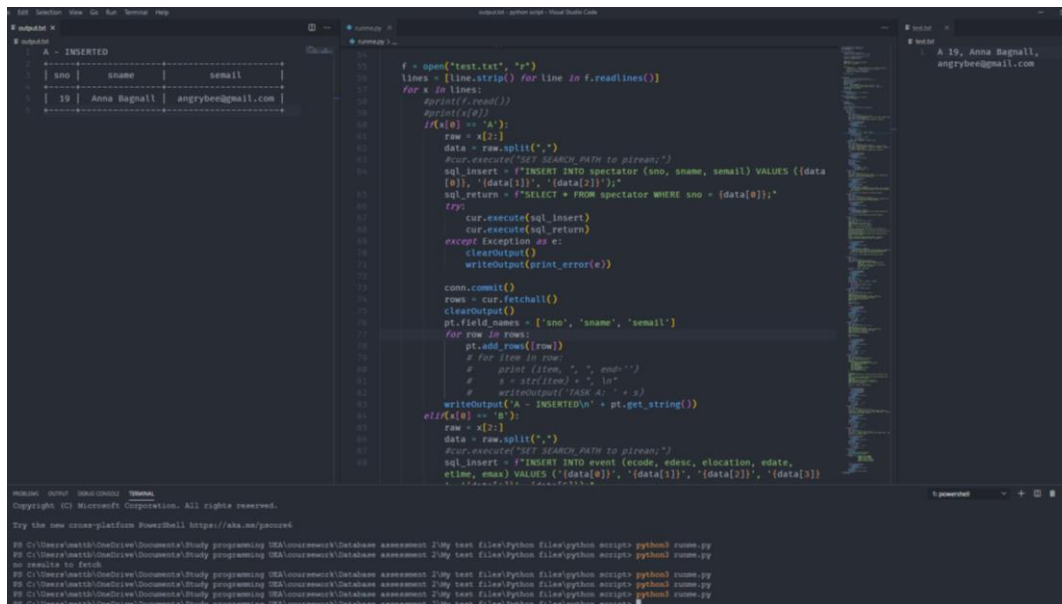
INPUT

sno	sname	semail
100	A. N.	ano@somewhere.net

```
lines = [line.strip() for line in f.readlines()]
for x in lines:
    #print(f.readline())
    #print(x[0])
    if(x[0] == 'A'):
        raw = x[2:]
        data = raw.split(",")
        #cur.execute("SET SEARCH_PATH to pirans;")
        sql_insert = "INSERT INTO spectator (sno, sname, seemail) VALUES ({data[0]}, '{data[1]}', '{data[2]}');"
        sql_return = "SELECT * FROM spectator WHERE sno = {data[0]};"
        try:
            cur.execute(sql_insert)
            cur.execute(sql_return)
        except Exception as e:
            clearOutput()
            writeOutput(print_error(e))

conn.commit()
rows = cur.fetchall()
clearOutput()
pt.field_names = ['sno', 'sname', 'seemail']
for row in rows:
    pt.add_rows([row])
    # for item in row:
    #     print(item, ", ", end='')
    #     s = str(item) + ", ln"
    #     writeOutput("TASK A: " + s)
writeOutput("A - INSERTED\n" + pt.get_string())
elif(x[0] == 'B'):
    raw = x[2:]
    data = raw.split(",")
    #cur.execute("SET SEARCH_PATH to pirans;")
    sql_insert = "INSERT INTO event (ecode, edesc, elocation, edate, etime, emax) VALUES ('{data[0]}', '{data[1]}', '{data[2]}', '{data[3]}', '{data[4]}', '{data[5]}');"
    sql_return = "SELECT * FROM event WHERE ecode = '{data[0]}';"
```

TEST 2 PASS A 19, Anna Bagnall, angrybee@gmail.com

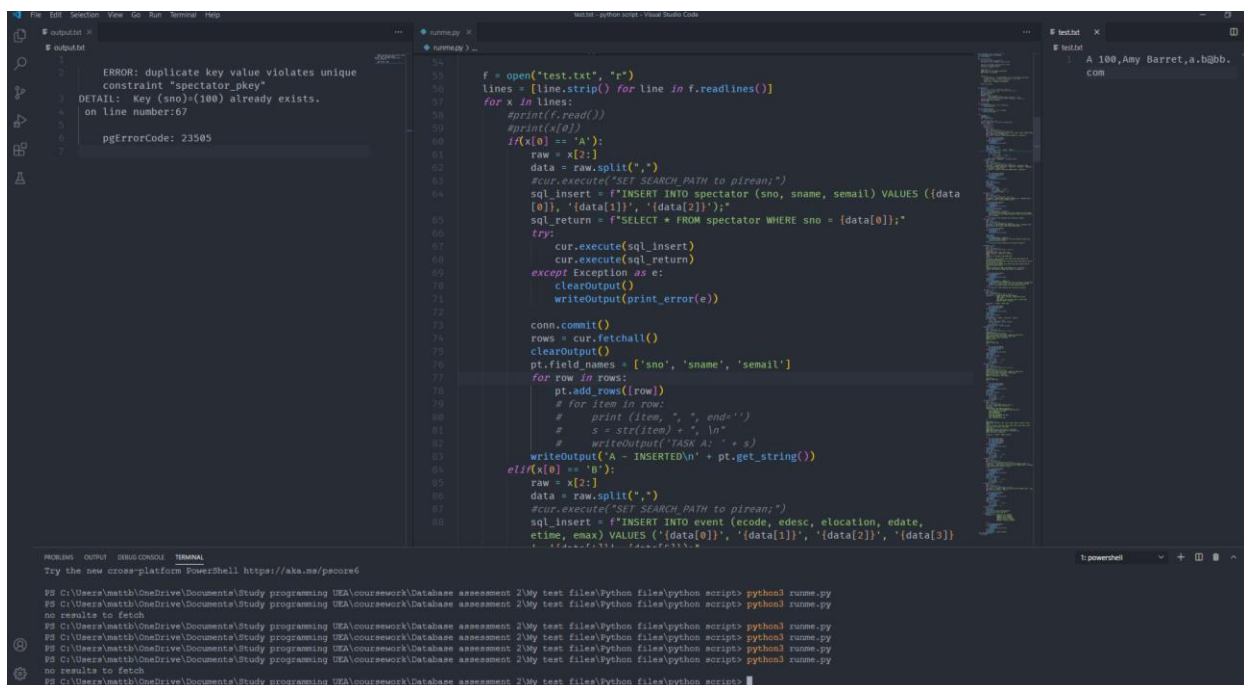


```
1 A - INSERTED
2 +-----+
3 | sno | sname | semail |
4 | 19 | Anna Bagnall | angrybee@gmail.com |
5 +-----+
```

```
54 f = open("test.txt", "r")
55 lines = [line.strip() for line in f.readlines()]
56 for x in lines:
57     #print(f.read())
58     #print(x[0])
59     raw = x[2:]
60     data = raw.split(",")
61     #cur.execute("SET SEARCH_PATH to p1rean;")
62     sql_insert = f"INSERT INTO spectator (sno, sname, semail) VALUES ({data
63     [0]}, '{data[1]}', '{data[2]}');"
64     sql_return = f"SELECT * FROM spectator WHERE sno = {data[0]};"
65     try:
66         cur.execute(sql_insert)
67         cur.execute(sql_return)
68     except Exception as e:
69         clearOutput()
70         writeOutput(print_error(e))
71
72 conn.commit()
73 rows = cur.fetchall()
74 clearOutput()
75 pt.field_names = ['sno', 'sname', 'semail']
76 for row in rows:
77     pt.add_rows([row])
78     # for item in row:
79     #     print(item, ", ", end='')
80     #     s = str(item) + ", in"
81     #     writeOutput("TASK A: " + s)
82 writeOutput("A - INSERTED\n" + pt.get_string())
83 elif(x[0] == "A"):
84     raw = x[2:]
85     data = raw.split(",")
86     #cur.execute("SET SEARCH_PATH to p1rean;")
87     sql_insert = f"INSERT INTO event (ecode, edesc, elocation, edate,
88     etime, emax) VALUES ('{data[0]}', '{data[1]}', '{data[2]}', '{data[3]}
89     ',"
```

output: A 19, Anna Bagnall, angrybee@gmail.com

TEST 3 FAIL (tno already exists) A 100,Amy Barret,a.b@bb.com



```
1 ERROR: duplicate key value violates unique
2 constraint "spectator_pkey"
3 DETAIL: Key (sno)=(100) already exists.
4 on line number:67
5 pgErrorcode: 23505
```

```
54 f = open("test.txt", "r")
55 lines = [line.strip() for line in f.readlines()]
56 for x in lines:
57     #print(f.read())
58     #print(x[0])
59     if(x[0] == 'A'):
60         raw = x[2:]
61         data = raw.split(",")
62         #cur.execute("SET SEARCH_PATH to p1rean;")
63         sql_insert = f"INSERT INTO spectator (sno, sname, semail) VALUES ({data
64         [0]}, '{data[1]}', '{data[2]}');"
65         sql_return = f"SELECT * FROM spectator WHERE sno = {data[0]};"
66         try:
67             cur.execute(sql_insert)
68             cur.execute(sql_return)
69         except Exception as e:
70             clearOutput()
71             writeOutput(print_error(e))
72
73 conn.commit()
74 rows = cur.fetchall()
75 clearOutput()
76 pt.field_names = ['sno', 'sname', 'semail']
77 for row in rows:
78     pt.add_rows([row])
79     # for item in row:
80     #     print(item, ", ", end='')
81     #     s = str(item) + ", in"
82     #     writeOutput("TASK A: " + s)
83 writeOutput("A - INSERTED\n" + pt.get_string())
84 elif(x[0] == 'B'):
85     raw = x[2:]
86     data = raw.split(",")
87     #cur.execute("SET SEARCH_PATH to p1rean;")
88     sql_insert = f"INSERT INTO event (ecode, edesc, elocation, edate,
89     etime, emax) VALUES ('{data[0]}', '{data[1]}', '{data[2]}', '{data[3]}
90     ',"
```

output: ERROR: duplicate key value violates unique constraint "spectator_pkey" DETAIL: Key (sno)=(100) already exists. on line number:67 pgErrorcode: 23505

Final spectator table for comparison:

```
pirean=# select * from spectator;
```

sno	sname	semail
2	Sam Bagnall	sam.bagnall@gmail.com
3	Anna Bagnall	anna.bagnall@gmail.com
4	Sarah Hodgson	sarah.hodgson@gmail.com
5	Dave	d@d.com
6	David	dav@fa.com
7	Dave	d@d.com
8	David	dav@fa.com
100	A. N.	ano@somewhere.net
19	Anna Bagnall	angrybee@gmail.com

(9 rows)

```
pirean=#
```

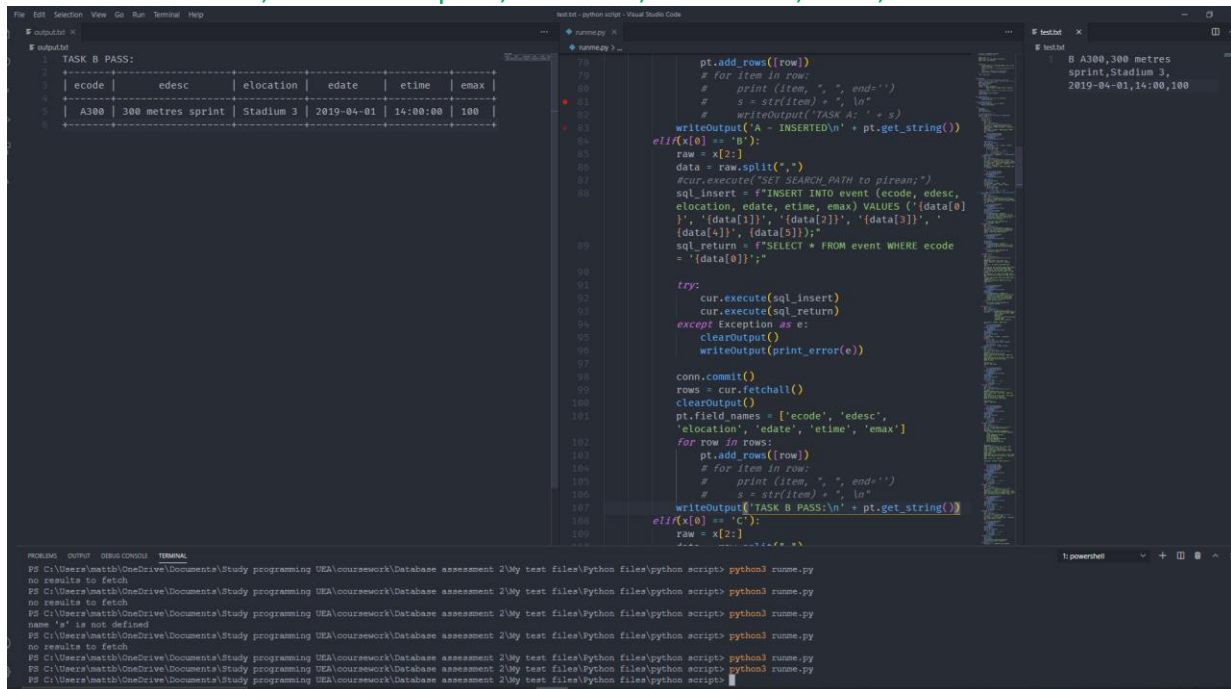
TAKS B)

TEST 1 PASS B A100,100 metres sprint,Stadium 1,2019-04-01,14:00,1000

```
1 TASK B PASS:
2 -----
3 |ecode| edesc | elocation | edate | etime | emax |
4 |-----|-----|-----|-----|-----|-----|
5 | A100 | 100 metres sprint | Stadium 1 | 2019-04-01 | 14:00:00 | 1000 |
6 |-----|-----|-----|-----|-----|-----|

7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
```

TEST 2 PASS B A300,300 metres sprint,Stadium 3,2019-04-01,14:00,100



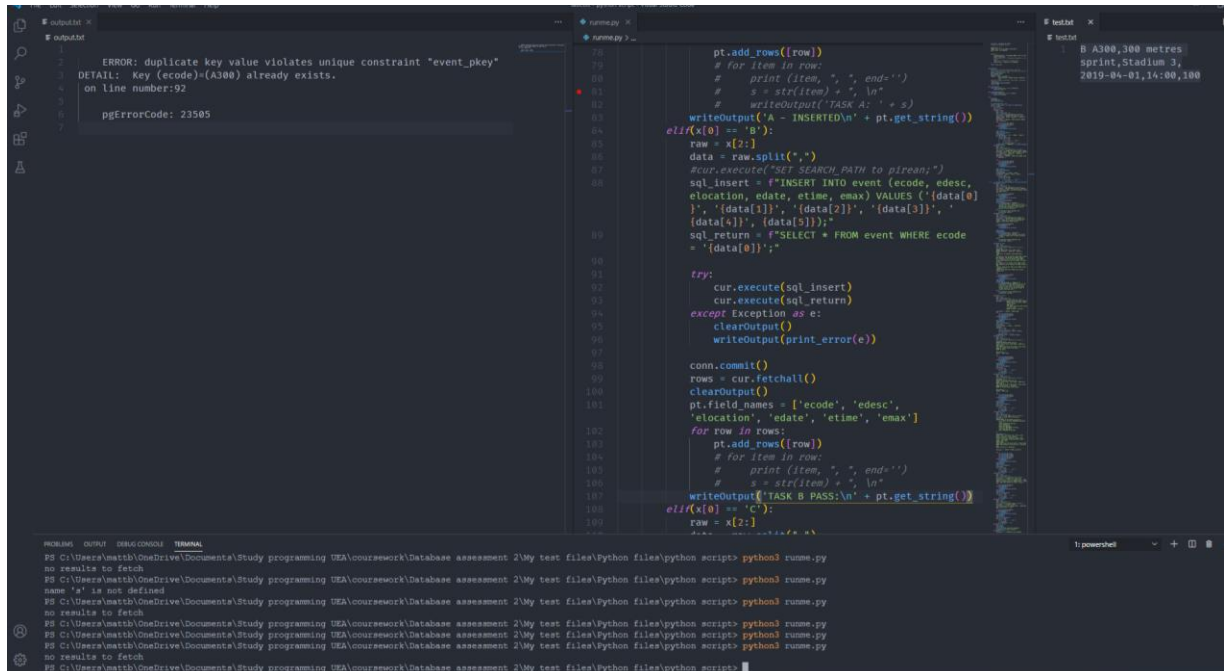
```
File Edit View Go Run Terminal Help
Python: Python Script - Visual Studio Code
TASK B PASS:
|-----|
| ecode | edesc | elocation | edate | etime | emax |
|-----|
| A300 | 300 metres sprint | Stadium 3 | 2019-04-01 | 14:00:00 | 100 |
|-----|

pt.add_rows([row])
# for item in row:
#     print(item, ", ", end='')
#     s = str(item) + ", "
#     writeOutput("TASK A: " + s)
writeOutput('A - INSERTED\n' + pt.get_string())
elif(x[0] == 'B'):
    row = x[2:]
    data = row.split(",")
    #cur.execute("SET SEARCH_PATH to p1rean;")
    sql_insert = f"INSERT INTO event (ecode, edesc,
    elocation, edate, etime, emax) VALUES ('{data[0]}',
    '{data[1]}', '{data[2]}', '{data[3]}', '{data[4]}', '{data[5]}');"
    sql_return = f"SELECT * FROM event WHERE ecode = '{data[0]}';"
    try:
        cur.execute(sql_insert)
        cur.execute(sql_return)
    except Exception as e:
        clearOutput()
        writeOutput(print_error(e))

    conn.commit()
    rows = cur.fetchall()
    clearOutput()
    pt.field_names = ['ecode', 'edesc',
    'elocation', 'edate', 'etime', 'emax']
    for row in rows:
        pt.add_rows([row])
        # for item in row:
        #     print(item, ", ", end='')
        #     s = str(item) + ", "
        #     writeOutput("TASK B PASS:\n" + pt.get_string())
    elif(x[0] == 'C'):
        row = x[2:]
        # for item in row:
        #     print(item, ", ", end='')
        #     s = str(item) + ", "
        #     writeOutput("TASK C PASS:\n" + pt.get_string())

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
PS C:\Users\math\OneDrive\Documents\Study programming\UEA\coursework\Database assessment 2\My test files\Python files\python script> python3 runme.py
no results to fetch
PS C:\Users\math\OneDrive\Documents\Study programming\UEA\coursework\Database assessment 2\My test files\Python files\python script> python3 runme.py
no results to fetch
PS C:\Users\math\OneDrive\Documents\Study programming\UEA\coursework\Database assessment 2\My test files\Python files\python script> python3 runme.py
name 's' is not defined
PS C:\Users\math\OneDrive\Documents\Study programming\UEA\coursework\Database assessment 2\My test files\Python files\python script> python3 runme.py
no results to fetch
PS C:\Users\math\OneDrive\Documents\Study programming\UEA\coursework\Database assessment 2\My test files\Python files\python script> python3 runme.py
PS C:\Users\math\OneDrive\Documents\Study programming\UEA\coursework\Database assessment 2\My test files\Python files\python script> python3 runme.py
PS C:\Users\math\OneDrive\Documents\Study programming\UEA\coursework\Database assessment 2\My test files\Python files\python script>
```

TEST 3 FAIL (Duplicate event) B A300,300 metres sprint,Stadium 3,2019-04-01,14:00,100



```
File Edit View Go Run Terminal Help
Python: Python Script - Visual Studio Code
ERROR: duplicate key value violates unique constraint "event_pkey"
DETAIL: Key (ecode)=(A300) already exists.
on line number:92
pgErrorcode: 23505

pt.add_rows([row])
# for item in row:
#     print(item, ", ", end='')
#     s = str(item) + ", "
#     writeOutput("TASK A: " + s)
writeOutput('A - INSERTED\n' + pt.get_string())
elif(x[0] == 'B'):
    row = x[2:]
    data = row.split(",")
    #cur.execute("SET SEARCH_PATH to p1rean;")
    sql_insert = f"INSERT INTO event (ecode, edesc,
    elocation, edate, etime, emax) VALUES ('{data[0]}',
    '{data[1]}', '{data[2]}', '{data[3]}', '{data[4]}', '{data[5]}');"
    sql_return = f"SELECT * FROM event WHERE ecode = '{data[0]}';"
    try:
        cur.execute(sql_insert)
        cur.execute(sql_return)
    except Exception as e:
        clearOutput()
        writeOutput(print_error(e))

    conn.commit()
    rows = cur.fetchall()
    clearOutput()
    pt.field_names = ['ecode', 'edesc',
    'elocation', 'edate', 'etime', 'emax']
    for row in rows:
        pt.add_rows([row])
        # for item in row:
        #     print(item, ", ", end='')
        #     s = str(item) + ", "
        #     writeOutput("TASK B PASS:\n" + pt.get_string())
    elif(x[0] == 'C'):
        row = x[2:]
        # for item in row:
        #     print(item, ", ", end='')
        #     s = str(item) + ", "
        #     writeOutput("TASK C PASS:\n" + pt.get_string())

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
PS C:\Users\math\OneDrive\Documents\Study programming\UEA\coursework\Database assessment 2\My test files\Python files\python script> python3 runme.py
no results to fetch
PS C:\Users\math\OneDrive\Documents\Study programming\UEA\coursework\Database assessment 2\My test files\Python files\python script> python3 runme.py
name 's' is not defined
PS C:\Users\math\OneDrive\Documents\Study programming\UEA\coursework\Database assessment 2\My test files\Python files\python script> python3 runme.py
no results to fetch
PS C:\Users\math\OneDrive\Documents\Study programming\UEA\coursework\Database assessment 2\My test files\Python files\python script> python3 runme.py
PS C:\Users\math\OneDrive\Documents\Study programming\UEA\coursework\Database assessment 2\My test files\Python files\python script> python3 runme.py
PS C:\Users\math\OneDrive\Documents\Study programming\UEA\coursework\Database assessment 2\My test files\Python files\python script> python3 runme.py
PS C:\Users\math\OneDrive\Documents\Study programming\UEA\coursework\Database assessment 2\My test files\Python files\python script>
```

Final event table for comparison:

```
pirean=# select * from event;
```

ecode	edesc	elocation	edate	etime	emax
S400	400m swimming event	London	2019-04-28	10:00:00	450
S600	400m swim	Stadium 1	2019-04-01	09:00:00	100
S900	400m swim	Stadium 1	2019-04-01	09:00:00	100
S200	400m swim	Stadium 1	2019-04-01	09:00:00	100
I54D	lorem id	Lodan Wetan	2019-04-10	17:55:00	91
U65A	imperdiet sapien	Perehonivka	2019-04-12	14:26:00	66
P6SL	dolor vel est donec	Trzcinnica	2019-04-06	11:03:00	62
I5SD	consequat metus	Bantarsari Wulon	2019-04-06	11:03:00	48
N4HL	et magnis	Beverwijk	2019-04-23	12:13:00	95
P54T	nonummy	Lapa do Lobo	2019-04-26	13:20:00	28
A100	100 metres sprint	Stadium 1	2019-04-01	14:00:00	1000
A300	300 metres sprint	Stadium 3	2019-04-01	14:00:00	100

(12 rows)

```
pirean=#
```

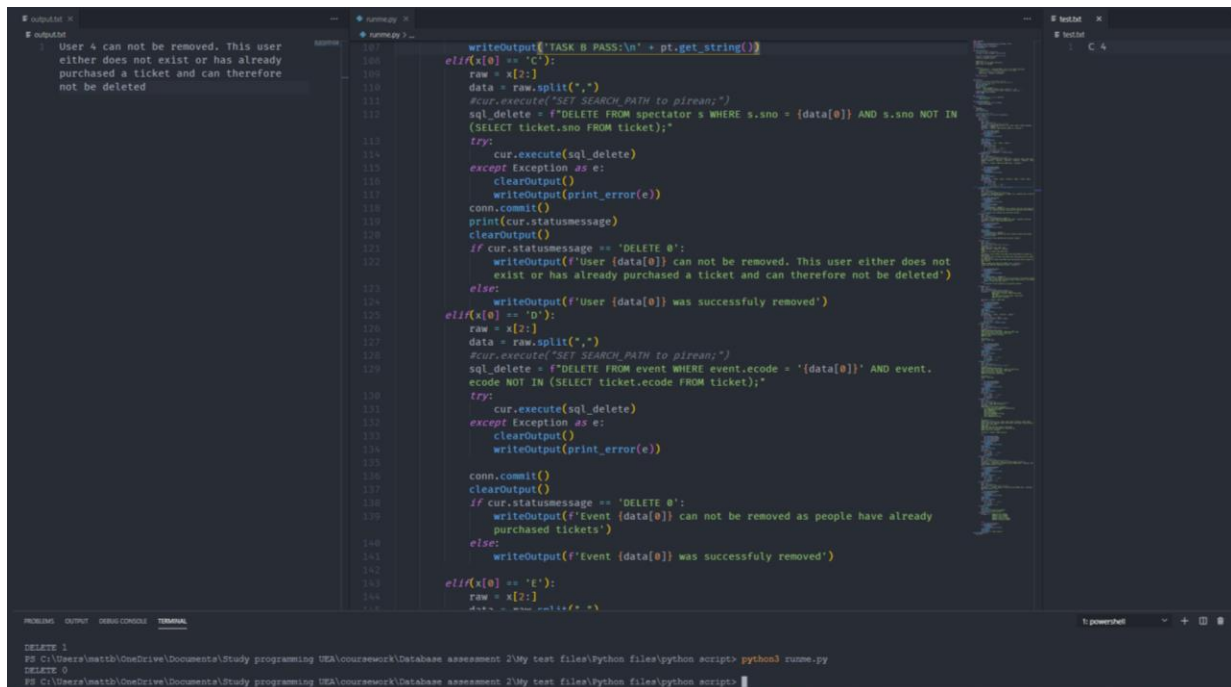
TAKS C)

TEST 1 PASS C 100 (Spectator does not have any tickets)

```
writeOutput('TASK B PASS:\n' + pt.get_string())
elif(x[0] == 'C'):
    raw = x[2:]
    data = raw.split(',')
    #cur.execute("SET SEARCH_PATH to pirean;")
    sql_delete = f"DELETE FROM spectator s WHERE s.sno = {data[0]} AND s.sno NOT IN (SELECT ticket.sno FROM ticket);"
    try:
        cur.execute(sql_delete)
    except Exception as e:
        clearOutput()
        writeOutput(print_error(e))
    conn.commit()
    print(cur.statusmessage)
    clearOutput()
    if cur.statusmessage == 'DELETE 0':
        writeOutput(f'User {data[0]} can not be removed. This user either does not exist or has already purchased a ticket and can therefore not be deleted')
    else:
        writeOutput(f'User {data[0]} was successfully removed')
elif(x[0] == 'D'):
    raw = x[2:]
    data = raw.split(',')
    #cur.execute("SET SEARCH_PATH to pirean;")
    sql_delete = f"DELETE FROM event WHERE event.ecode = '{data[0]}' AND event.ecode NOT IN (SELECT ticket.ecode FROM ticket);"
    try:
        cur.execute(sql_delete)
    except Exception as e:
        clearOutput()
        writeOutput(print_error(e))
    conn.commit()
    clearOutput()
    if cur.statusmessage == 'DELETE 0':
        writeOutput(f'Event {data[0]} can not be removed as people have already purchased tickets')
    else:
        writeOutput(f'Event {data[0]} was successfully removed')
elif(x[0] == 'E'):
    raw = x[2:]
    data = raw.split(',')
    #cur.execute("SET SEARCH_PATH to pirean;")
```

no results to fetch
PS C:\Users\umath\OneDrive\Documents\Study programming\UEA\coursework\Database assessment 2\My test files\python files\python script> python3 runme.py
DELETE 1
PS C:\Users\umath\OneDrive\Documents\Study programming\UEA\coursework\Database assessment 2\My test files\python files\python script>

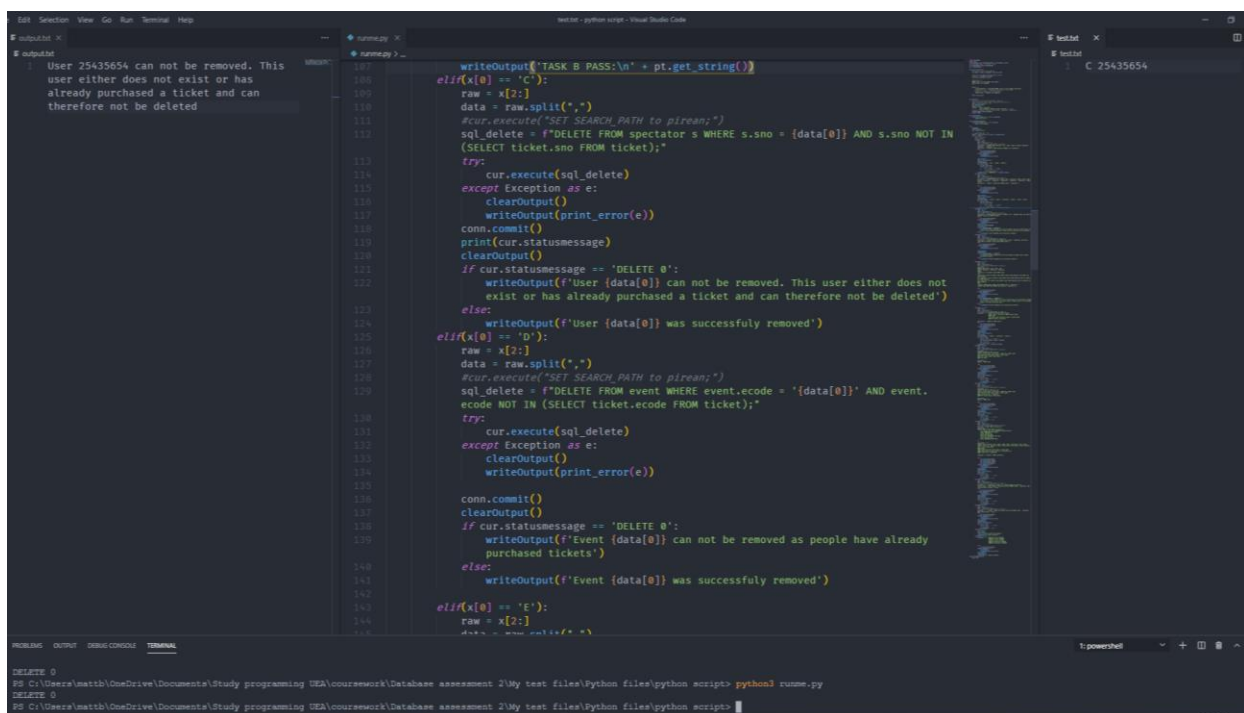
TEST 2 FAIL C 4 (Spectator has already purchased tickets so can not be deleted)



```
117 writeOutput("TASK B PASS:\n" + pt.get_string())
118 elif(x[0] == 'C'):
119     raw = x[2:]
120     data = raw.split(",")
121     #cur.execute("SET SEARCH_PATH to p1rean;")
122     sql_delete = f"DELETE FROM spectator s WHERE s.sno = {data[0]} AND s.sno NOT IN
123     (SELECT ticket.sno FROM ticket);"
124     try:
125         cur.execute(sql_delete)
126     except Exception as e:
127         clearOutput()
128         writeOutput(print_error(e))
129         conn.commit()
130         print(cur.statusmessage)
131         clearOutput()
132         if cur.statusmessage == 'DELETE 0':
133             writeOutput(f'User {data[0]} can not be removed. This user either does not
134             exist or has already purchased a ticket and can therefore not be deleted')
135         else:
136             writeOutput(f'User {data[0]} was successfully removed')
137 elif(x[0] == 'D'):
138     raw = x[2:]
139     data = raw.split(",")
140     #cur.execute("SET SEARCH_PATH to p1rean;")
141     sql_delete = f"DELETE FROM event WHERE event.ecode = '{data[0]}' AND event.
142     ecode NOT IN (SELECT ticket.ecode FROM ticket);"
143     try:
144         cur.execute(sql_delete)
145     except Exception as e:
146         clearOutput()
147         writeOutput(print_error(e))
148         conn.commit()
149         clearOutput()
150         if cur.statusmessage == 'DELETE 0':
151             writeOutput(f'Event {data[0]} can not be removed as people have already
152             purchased tickets')
153         else:
154             writeOutput(f'Event {data[0]} was successfully removed')
155 elif(x[0] == 'E'):
156     raw = x[2:]
157     data = raw.split(",")
```

DELETE 1
PS C:\Users\math\OneDrive\Documents\Study programming\UEA\coursework\Database assessment 2\My test files\Python files\python script> python3 runme.py
DELETE 0
PS C:\Users\math\OneDrive\Documents\Study programming\UEA\coursework\Database assessment 2\My test files\Python files\python script>

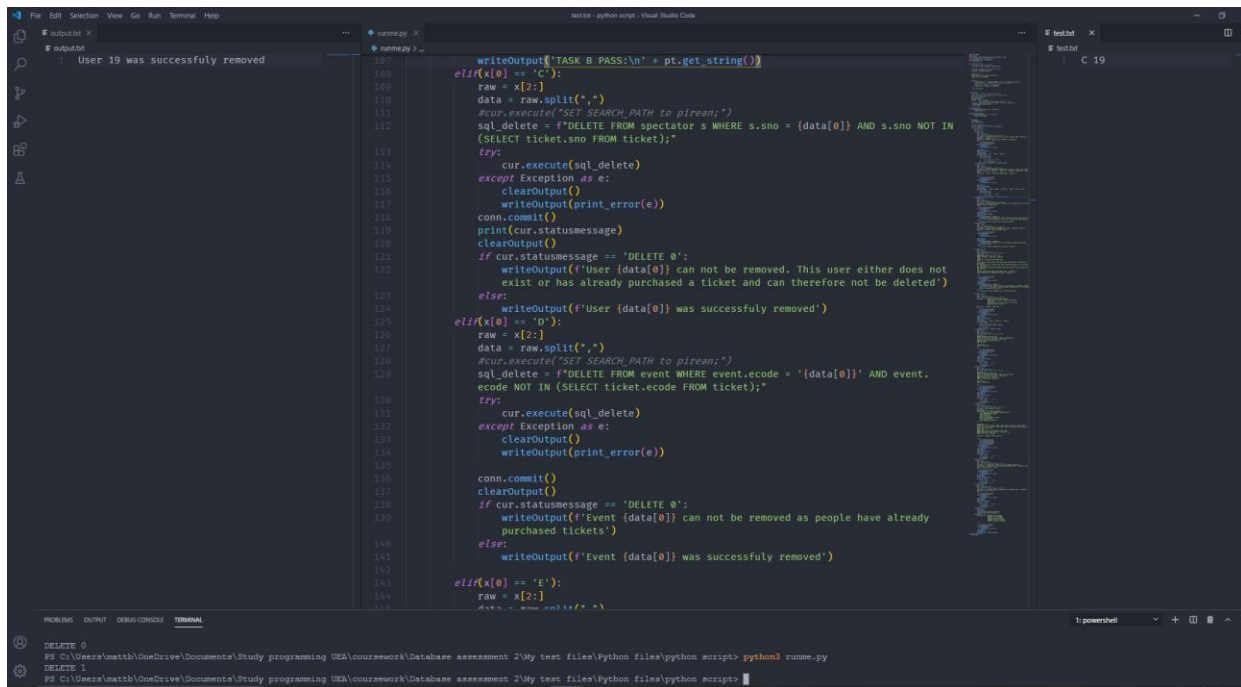
TEST 3 FAIL C 25435654 (spectator does not exist)



```
117 writeOutput("TASK B PASS:\n" + pt.get_string())
118 elif(x[0] == 'C'):
119     raw = x[2:]
120     data = raw.split(",")
121     #cur.execute("SET SEARCH_PATH to p1rean;")
122     sql_delete = f"DELETE FROM spectator s WHERE s.sno = {data[0]} AND s.sno NOT IN
123     (SELECT ticket.sno FROM ticket);"
124     try:
125         cur.execute(sql_delete)
126     except Exception as e:
127         clearOutput()
128         writeOutput(print_error(e))
129         conn.commit()
130         print(cur.statusmessage)
131         clearOutput()
132         if cur.statusmessage == 'DELETE 0':
133             writeOutput(f'User {data[0]} can not be removed. This user either does not
134             exist or has already purchased a ticket and can therefore not be deleted')
135         else:
136             writeOutput(f'User {data[0]} was successfully removed')
137 elif(x[0] == 'D'):
138     raw = x[2:]
139     data = raw.split(",")
140     #cur.execute("SET SEARCH_PATH to p1rean;")
141     sql_delete = f"DELETE FROM event WHERE event.ecode = '{data[0]}' AND event.
142     ecode NOT IN (SELECT ticket.ecode FROM ticket);"
143     try:
144         cur.execute(sql_delete)
145     except Exception as e:
146         clearOutput()
147         writeOutput(print_error(e))
148         conn.commit()
149         clearOutput()
150         if cur.statusmessage == 'DELETE 0':
151             writeOutput(f'Event {data[0]} can not be removed as people have already
152             purchased tickets')
153         else:
154             writeOutput(f'Event {data[0]} was successfully removed')
155 elif(x[0] == 'E'):
156     raw = x[2:]
157     data = raw.split(",")
```

DELETE 0
PS C:\Users\math\OneDrive\Documents\Study programming\UEA\coursework\Database assessment 2\My test files\Python files\python script> python3 runme.py
DELETE 0
PS C:\Users\math\OneDrive\Documents\Study programming\UEA\coursework\Database assessment 2\My test files\Python files\python script>

TEST 4 PASS C 19 (spectator does not have any tickets)



```
107 writeOutput("TASK B PASS:\n" + pt.get_string())
108
109 elif(x[0] == 'C'):
110     raw = x[2:]
111     data = raw.split(",")
112     #cur.execute("SET SEARCH_PATH to pirean;")
113     sql_delete = f"DELETE FROM spectator s WHERE s.sno = {data[0]} AND s.sno NOT IN
114     (SELECT ticket.sno FROM ticket);"
115     try:
116         cur.execute(sql_delete)
117     except Exception as e:
118         clearOutput()
119         writeOutput(print_error(e))
120     conn.commit()
121     print(cur.statusmessage)
122     clearOutput()
123     if cur.statusmessage == 'DELETE 0':
124         writeOutput(f'User {data[0]} can not be removed. This user either does not
125         exist or has already purchased a ticket and can therefore not be deleted')
126     else:
127         writeOutput(f'User {data[0]} was successfully removed')
128
129 elif(x[0] == 'D'):
130     raw = x[2:]
131     data = raw.split(",")
132     #cur.execute("SET SEARCH_PATH to pirean;")
133     sql_delete = f"DELETE FROM event WHERE event.ecode = '{data[0]}' AND event.
134     ecode NOT IN (SELECT ticket.ecode FROM ticket);"
135     try:
136         cur.execute(sql_delete)
137     except Exception as e:
138         clearOutput()
139         writeOutput(print_error(e))
140     conn.commit()
141     clearOutput()
142     if cur.statusmessage == 'DELETE 0':
143         writeOutput(f'Event {data[0]} can not be removed as people have already
144         purchased tickets')
145     else:
146         writeOutput(f'Event {data[0]} was successfully removed')
147
148 elif(x[0] == 'E'):
149     raw = x[2:]
150     data = raw.split(",")
```

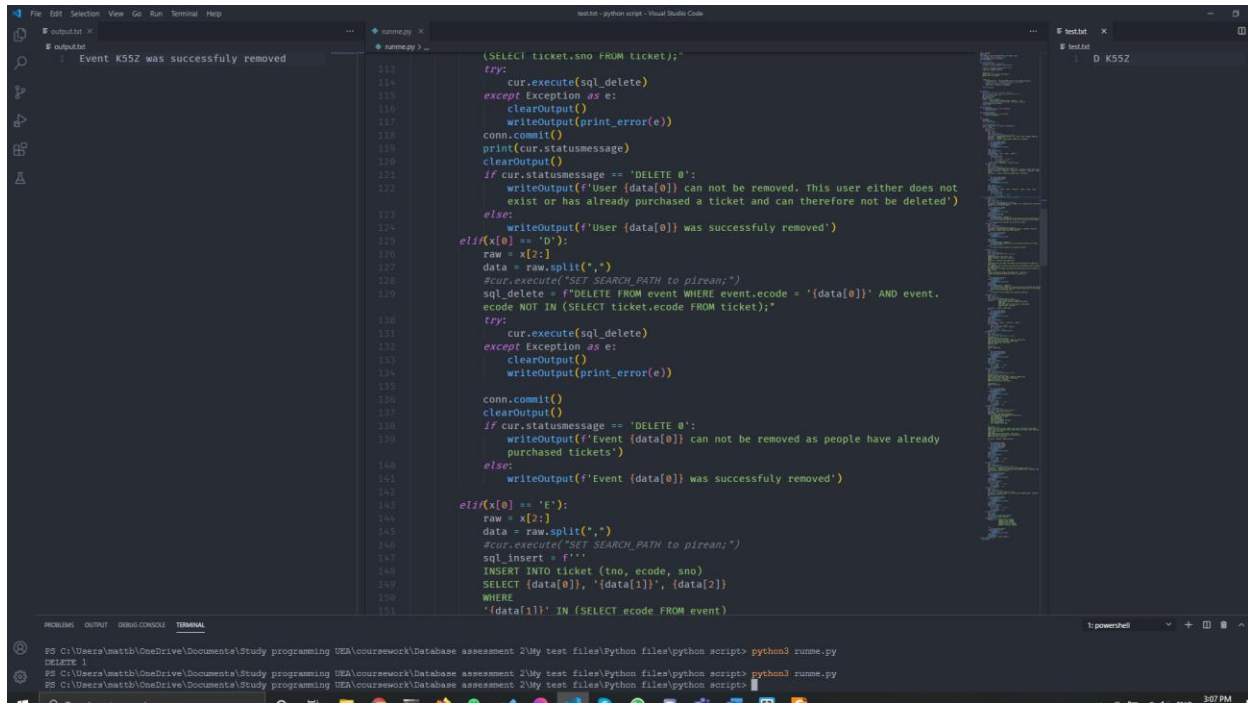
Final spectator table for proof of deletion. Remember that we deleted spectators that were added in task A, so the table has returned to normal.

```
pirean=# select * from spectator;
 sno |      sname      |      semail
-----+-----+-----
  2 | Sam Bagnall     | sam.bagnall@gmail.com
  3 | Anna Bagnall    | anna.bagnall@gmail.com
  4 | Sarah Hodgson   | sarah.hodgson@gmail.com
  5 | Dave            | d@d.com
  6 | David           | dav@fa.com
  7 | Dave            | d@d.com
  8 | David           | dav@fa.com
(7 rows)

pirean=#
```

TAKS D)

TEST 1 PASS D K55Z (Event can be deleted as no tickets have been sold to the event)

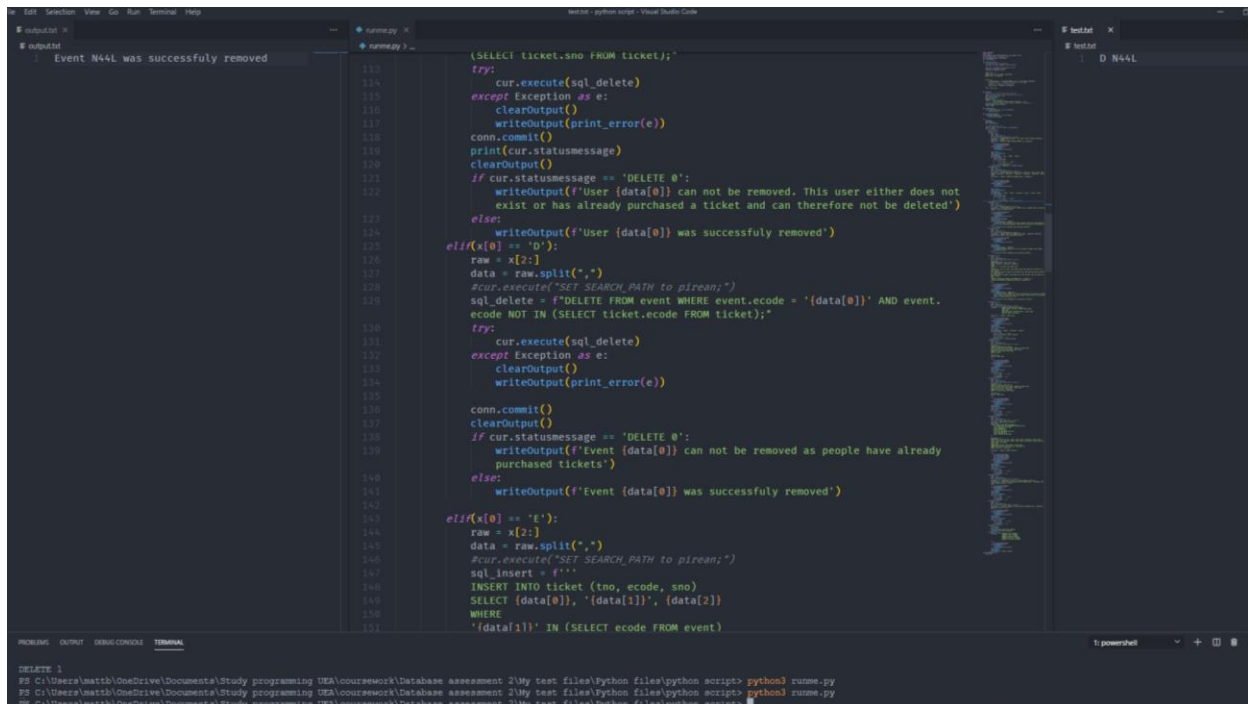


```
113 (SELECT ticket.sno FROM ticket);"
114 try:
115     cur.execute(sql_delete)
116 except Exception as e:
117     clearOutput()
118     writeOutput(print_error(e))
119     conn.commit()
120     print(cur.statusmessage)
121     clearOutput()
122     if cur.statusmessage == 'DELETE 0':
123         writeOutput(f'User {data[0]} can not be removed. This user either does not
124         exist or has already purchased a ticket and can therefore not be deleted')
125     else:
126         writeOutput(f'User {data[0]} was successfully removed')
127 elif(x[0] == 'D'):
128     raw = x[2:]
129     data = raw.split(',')
130     #cur.execute("SET SEARCH_PATH to p1rean;")
131     sql_delete = f"DELETE FROM event WHERE event.ecode = '{data[0]}' AND event.
132     ecode NOT IN (SELECT ticket.ecode FROM ticket);"
133     try:
134         cur.execute(sql_delete)
135     except Exception as e:
136         clearOutput()
137         writeOutput(print_error(e))
138     conn.commit()
139     clearOutput()
140     if cur.statusmessage == 'DELETE 0':
141         writeOutput(f'Event {data[0]} can not be removed as people have already
142         purchased tickets')
143     else:
144         writeOutput(f'Event {data[0]} was successfully removed')
145 elif(x[0] == 'E'):
146     raw = x[2:]
147     data = raw.split(',')
148     #cur.execute("SET SEARCH_PATH to p1rean;")
149     sql_insert = f'''
150     INSERT INTO ticket (tno, ecode, sno)
151     SELECT {data[0]}, '{data[1]}', {data[2]}
152     WHERE
153     '{data[1]}' IN (SELECT ecode FROM event)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

PS C:\Users\matth\OneDrive\Documents\Study programming\UEA\coursework\Database assessment 2\My test files\Python files\python scripts> python3 runme.py
Delete 1
PS C:\Users\matth\OneDrive\Documents\Study programming\UEA\coursework\Database assessment 2\My test files\Python files\python scripts> python3 runme.py
PS C:\Users\matth\OneDrive\Documents\Study programming\UEA\coursework\Database assessment 2\My test files\Python files\python scripts>

TEST 2 PASS D N44L (No tickets have ben sold for this event)

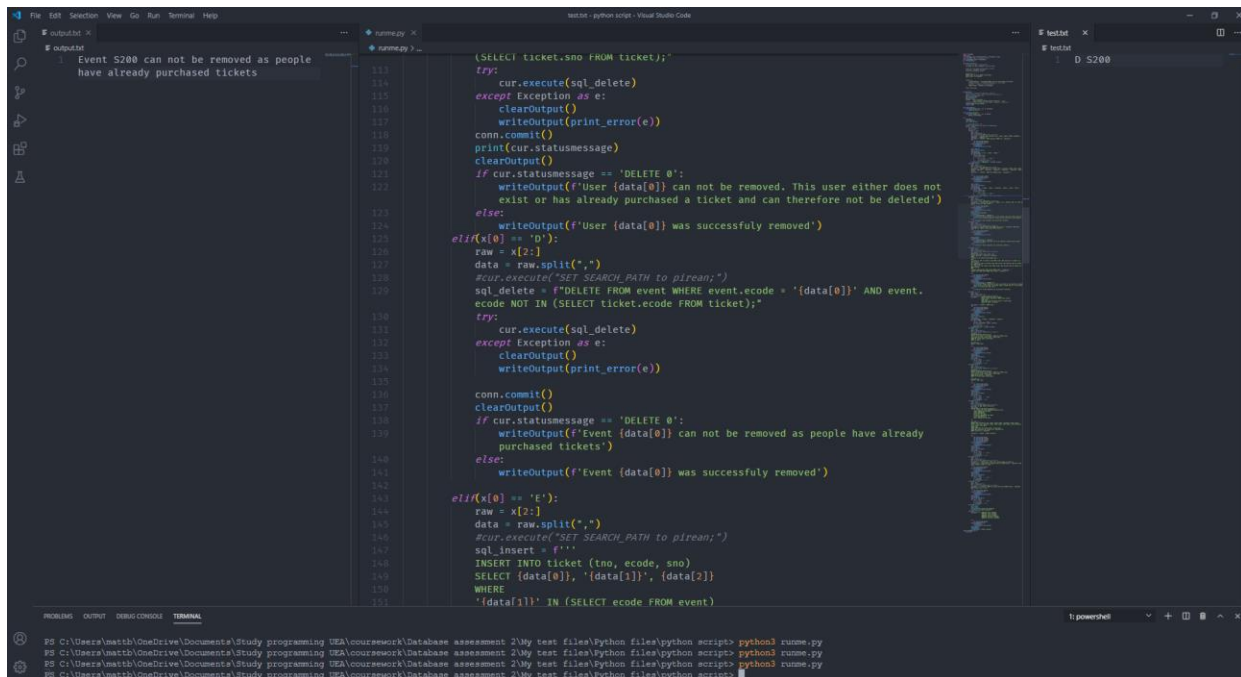


```
113 (SELECT ticket.sno FROM ticket);"
114 try:
115     cur.execute(sql_delete)
116 except Exception as e:
117     clearOutput()
118     writeOutput(print_error(e))
119     conn.commit()
120     print(cur.statusmessage)
121     clearOutput()
122     if cur.statusmessage == 'DELETE 0':
123         writeOutput(f'User {data[0]} can not be removed. This user either does not
124         exist or has already purchased a ticket and can therefore not be deleted')
125     else:
126         writeOutput(f'User {data[0]} was successfully removed')
127 elif(x[0] == 'D'):
128     raw = x[2:]
129     data = raw.split(',')
130     #cur.execute("SET SEARCH_PATH to p1rean;")
131     sql_delete = f"DELETE FROM event WHERE event.ecode = '{data[0]}' AND event.
132     ecode NOT IN (SELECT ticket.ecode FROM ticket);"
133     try:
134         cur.execute(sql_delete)
135     except Exception as e:
136         clearOutput()
137         writeOutput(print_error(e))
138     conn.commit()
139     clearOutput()
140     if cur.statusmessage == 'DELETE 0':
141         writeOutput(f'Event {data[0]} can not be removed as people have already
142         purchased tickets')
143     else:
144         writeOutput(f'Event {data[0]} was successfully removed')
145 elif(x[0] == 'E'):
146     raw = x[2:]
147     data = raw.split(',')
148     #cur.execute("SET SEARCH_PATH to p1rean;")
149     sql_insert = f'''
150     INSERT INTO ticket (tno, ecode, sno)
151     SELECT {data[0]}, '{data[1]}', {data[2]}
152     WHERE
153     '{data[1]}' IN (SELECT ecode FROM event)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

DELETE 1
PS C:\Users\matth\OneDrive\Documents\Study programming\UEA\coursework\Database assessment 2\My test files\Python files\python scripts> python3 runme.py
PS C:\Users\matth\OneDrive\Documents\Study programming\UEA\coursework\Database assessment 2\My test files\Python files\python scripts> python3 runme.py
PS C:\Users\matth\OneDrive\Documents\Study programming\UEA\coursework\Database assessment 2\My test files\Python files\python scripts>

TEST 3 FAIL D S200 (Tickets have been sold to the event already)



```
(SELECT ticket.sno FROM ticket);"
try:
    cur.execute(sql_delete)
except Exception as e:
    clearOutput()
    writeOutput(print_error(e))
conn.commit()
print(cur.statusmessage)
clearOutput()
if cur.statusmessage == 'DELETE 0':
    writeOutput(f'User {data[0]} can not be removed. This user either does not exist or has already purchased a ticket and can therefore not be deleted')
else:
    writeOutput(f'User {data[0]} was successfully removed')
elif(x[0] == 'D'):
    raw = x[2:]
    data = raw.split(",")
    #cur.execute("SET SEARCH_PATH to p1rean;")
    sql_delete = f"DELETE FROM event WHERE event.ecode = '{data[0]}' AND event.ecode NOT IN (SELECT ticket.ecode FROM ticket);"
    try:
        cur.execute(sql_delete)
    except Exception as e:
        clearOutput()
        writeOutput(print_error(e))
    conn.commit()
    clearOutput()
    if cur.statusmessage == 'DELETE 0':
        writeOutput(f'Event {data[0]} can not be removed as people have already purchased tickets')
    else:
        writeOutput(f'Event {data[0]} was successfully removed')
elif(x[0] == 'E'):
    raw = x[2:]
    data = raw.split(",")
    #cur.execute("SET SEARCH_PATH to p1rean;")
    sql_insert = f'''
INSERT INTO ticket (tmo, ecode, sno)
SELECT {data[0]}, '{data[1]}', {data[2]}
WHERE
'{data[1]}' IN (SELECT ecode FROM event)
```

Final tables for comprison:

```
pirean=# select * from event;
```

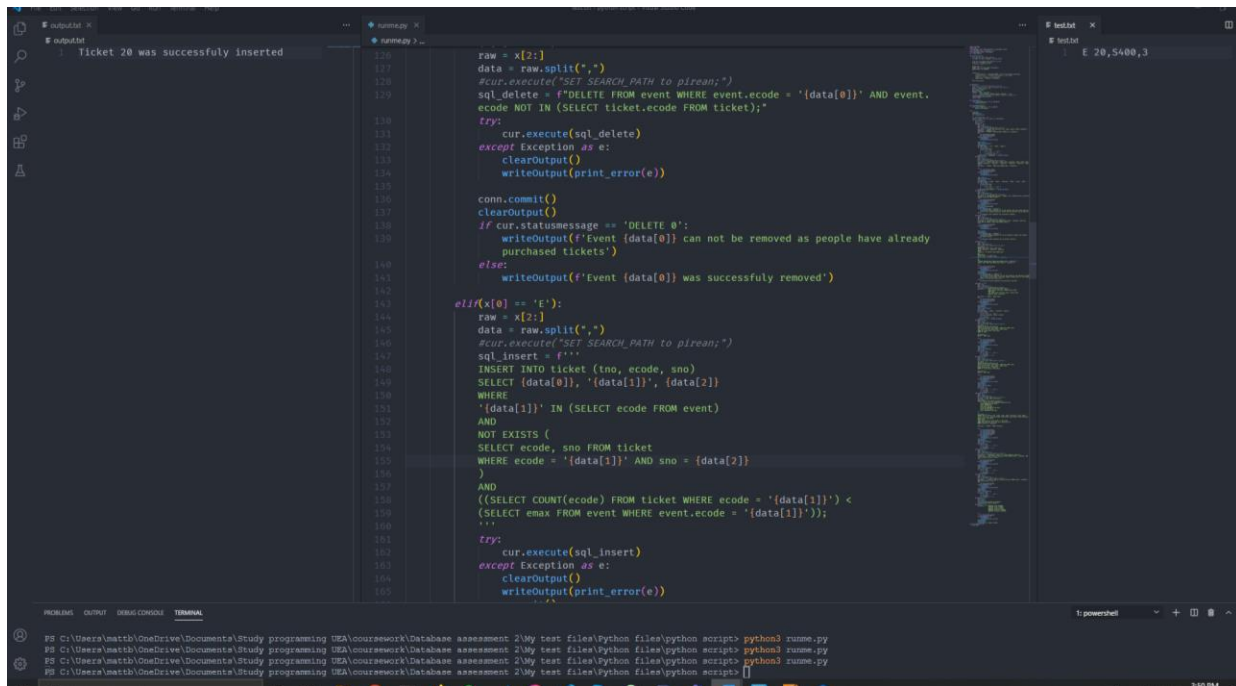
ecode	edesc	elocation	edate	etime	emax
S400	400m swimming event	London	2019-04-28	10:00:00	450
S600	400m swim	Stadium 1	2019-04-01	09:00:00	100
S900	400m swim	Stadium 1	2019-04-01	09:00:00	100
S200	400m swim	Stadium 1	2019-04-01	09:00:00	100
I54D	lorem id	Lodan Wetan	2019-04-10	17:55:00	91
U55A	imperdiet sapien	Perehonivka	2019-04-12	14:26:00	66
P55L	dolor vel est donec	Trzcinica	2019-04-06	11:03:00	62
I55D	consequat metus	Bantarsari Kulon	2019-04-06	11:03:00	48
P54T	nonummy	Lapa do Lobo	2019-04-26	13:20:00	28
A100	100 metres sprint	Stadium 1	2019-04-01	14:00:00	1000
A300	300 metres sprint	Stadium 3	2019-04-01	14:00:00	100

(11 rows)

```
pirean=#
```

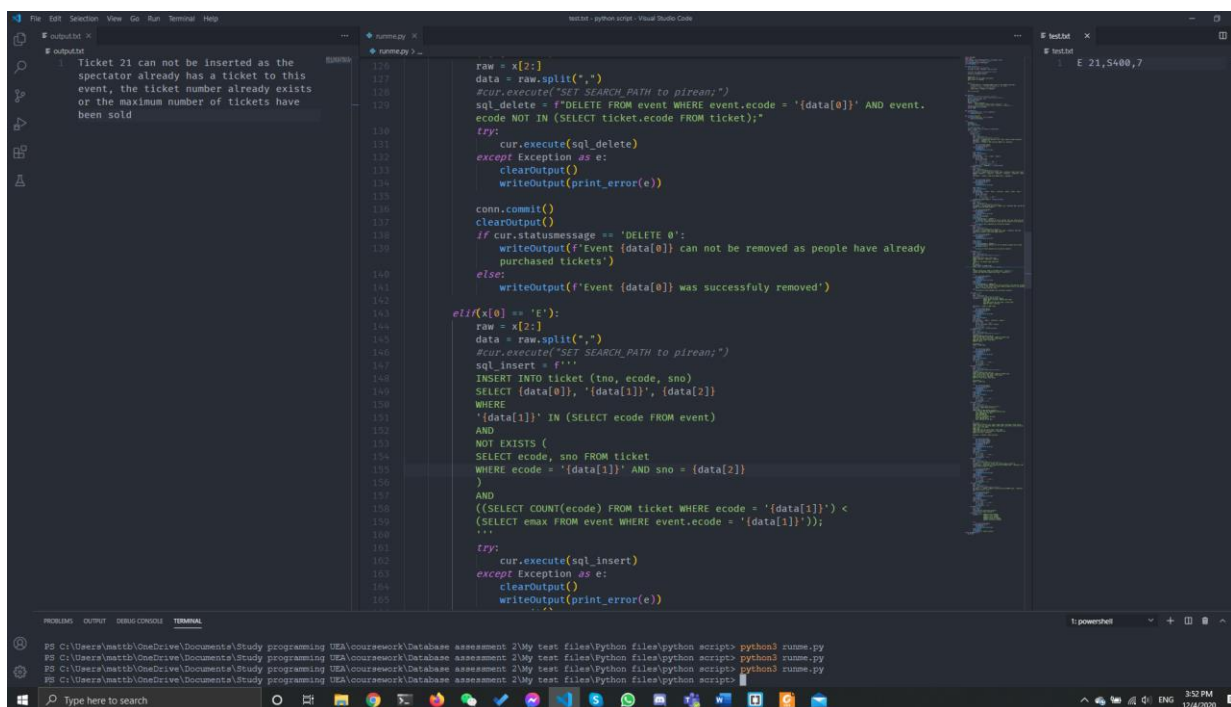
TASK E

TEST 1 PASS E 20,\$400,3 (Event exists and customer does not already have a ticket to that event)



```
126 row = x[2:]
127 data = raw.split(",")
128 #cur.execute("SET SEARCH_PATH to p1rean;")
129 sql_delete = f"DELETE FROM event WHERE event.ecode = '{data[0]}' AND event.
130 ecode NOT IN (SELECT ticket.ecode FROM ticket);"
131 try:
132     cur.execute(sql_delete)
133 except Exception as e:
134     clearOutput()
135     writeOutput(print_error(e))
136
137 conn.commit()
138 clearOutput()
139 if cur.statusmessage == 'DELETE 0':
140     writeOutput(f'Event {data[0]} can not be removed as people have already
141     purchased tickets')
142 else:
143     writeOutput(f'Event {data[0]} was successfully removed')
144
145 elif(x[0] == 'E'):
146     raw = x[2:]
147     data = raw.split(",")
148     #cur.execute("SET SEARCH_PATH to p1rean;")
149     sql_insert = f'''
150     INSERT INTO ticket (tno, ecode, sno)
151     SELECT {data[0]}, '{data[1]}', {data[2]}
152     WHERE
153     '{data[1]}' IN (SELECT ecode FROM event)
154     AND
155     NOT EXISTS (
156     SELECT ecode, sno FROM ticket
157     WHERE ecode = '{data[1]}' AND sno = {data[2]}
158     )
159     AND
160     ((SELECT COUNT(ecode) FROM ticket WHERE ecode = '{data[1]}') <
161     (SELECT emax FROM event WHERE event.ecode = '{data[1]}'));
162     '''
163     try:
164         cur.execute(sql_insert)
165     except Exception as e:
166         clearOutput()
167         writeOutput(print_error(e))
168     .
```

TEST 2 FAIL E 21, \$400, 7 (Event exists but customer already has a ticket to this event)



```
126 row = x[2:]
127 data = raw.split(",")
128 #cur.execute("SET SEARCH_PATH to p1rean;")
129 sql_delete = f"DELETE FROM event WHERE event.ecode = '{data[0]}' AND event.
130 ecode NOT IN (SELECT ticket.ecode FROM ticket);"
131 try:
132     cur.execute(sql_delete)
133 except Exception as e:
134     clearOutput()
135     writeOutput(print_error(e))
136
137 conn.commit()
138 clearOutput()
139 if cur.statusmessage == 'DELETE 0':
140     writeOutput(f'Event {data[0]} can not be removed as people have already
141     purchased tickets')
142 else:
143     writeOutput(f'Event {data[0]} was successfully removed')
144
145 elif(x[0] == 'E'):
146     raw = x[2:]
147     data = raw.split(",")
148     #cur.execute("SET SEARCH_PATH to p1rean;")
149     sql_insert = f'''
150     INSERT INTO ticket (tno, ecode, sno)
151     SELECT {data[0]}, '{data[1]}', {data[2]}
152     WHERE
153     '{data[1]}' IN (SELECT ecode FROM event)
154     AND
155     NOT EXISTS (
156     SELECT ecode, sno FROM ticket
157     WHERE ecode = '{data[1]}' AND sno = {data[2]}
158     )
159     AND
160     ((SELECT COUNT(ecode) FROM ticket WHERE ecode = '{data[1]}') <
161     (SELECT emax FROM event WHERE event.ecode = '{data[1]}'));
162     '''
163     try:
164         cur.execute(sql_insert)
165     except Exception as e:
166         clearOutput()
167         writeOutput(print_error(e))
168     .
```

TEST 3 FAIL E 23, SZZZ, 6 (event does not exist so a ticket can not be issued)

```
126 raw = x[2:]
127 data = raw.split(",")
128 #cur.execute("SET SEARCH_PATH to pirean;")
129 sql_delete = f"DELETE FROM event WHERE event.ecode = '{data[0]}' AND event.
130 ecode NOT IN (SELECT ticket.ecode FROM ticket);"
131 try:
132     cur.execute(sql_delete)
133 except Exception as e:
134     clearOutput()
135     writeOutput(print_error(e))
136
137 conn.commit()
138 clearOutput()
139 if cur.statusmessage == 'DELETE 0':
140     writeOutput(f'Event {data[0]} can not be removed as people have already
141     purchased tickets')
142 else:
143     writeOutput(f'Event {data[0]} was successfully removed')
144
145 elif(x[0] == 'E'):
146     raw = x[2:]
147     data = raw.split(",")
148     #cur.execute("SET SEARCH_PATH to pirean;")
149     sql_insert = f'''
150     INSERT INTO ticket (tno, ecode, sno)
151     SELECT {data[0]}, '{data[1]}', {data[2]}
152     WHERE
153     '{data[1]}' IN (SELECT ecode FROM event)
154     AND
155     NOT EXISTS (
156     SELECT ecode, sno FROM ticket
157     WHERE ecode = '{data[1]}' AND sno = {data[2]}
158     )
159     AND
160     ((SELECT COUNT(ecode) FROM ticket WHERE ecode = '{data[1]}') <
161     (SELECT emax FROM event WHERE event.ecode = '{data[1]}'));
162     '''
163     try:
164         cur.execute(sql_insert)
165     except Exception as e:
166         clearOutput()
167         writeOutput(print_error(e))
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
```

Final ticket table

```
pirean=# select * from ticket;
 tno | ecode | sno
-----+-----+-----
  2 | S200 | 2
  3 | S400 | 4
  4 | S600 | 4
  5 | S600 | 3
  6 | P55L | 3
  7 | P55L | 5
  9 | S400 | 7
 10 | P55L | 7
 11 | S600 | 8
 12 | S600 | 6
 13 | P55L | 6
 20 | S400 | 3
(12 rows)
```

TASK P:

```

1  P
2  |-----|
3  | edate | elocation | amount |
4  |-----|
5  | 2019-04-06 | Bantarsari Kulon | 0 |
6  | 2019-04-10 | Lodon Wetan | 0 |
7  | 2019-04-06 | Trzcina | 4 |
8  | 2019-04-01 | Stadium 3 | 0 |
9  | 2019-04-26 | Lapa do Lobo | 0 |
10 | 2019-04-12 | Perehoniwka | 0 |
11 | 2019-04-28 | London | 3 |
12 | 2019-04-01 | Stadium 1 | 5 |
13 |-----|

156 conn.commit()
157 clearOutput()
158 if cur.statusmessage == 'INSERT 0 0':
159     writeOutput(f'Ticket {data[0]} can not be inserted as the
160     spectator already has a ticket to this event, the ticket number
161     already exists or the maximum number of tickets have been sold')
162 else:
163     writeOutput(f'Ticket {data[0]} was successfully inserted')
164
165 elif(x[0] == 'P'):
166     raw = x[2:]
167     data = raw.split(",")
168     #cur.execute("SET SEARCH_PATH to piraan;")
169     sql_query = f'''CREATE OR REPLACE VIEW viewp AS
170     SELECT edate, elocation, COUNT(ticket.ecode)
171     FROM event
172     LEFT JOIN ticket ON event.ecode = ticket.ecode
173     GROUP BY edate, elocation;
174
175     sql_return = f'SELECT * FROM viewp;'
176     try:
177         cur.execute(sql_query)
178         cur.execute(sql_return)
179     except Exception as e:
180         clearOutput()
181         writeOutput(print_error(e))
182         conn.commit()
183         clearOutput()
184         rows = cur.fetchall()
185         print(rows)
186         pt.field_names = ['edate', 'elocation', 'amount']
187         for row in rows:
188             #print(item, ", ", end='')
189             pt.add_row([row[0], row[1], row[2]])
190             #s = str(row)
191             #writeOutput(s + '\n')
192         writeOutput('P\n' + pt.get_string())
193 elif(x[0] == 'Q'):
194     raw = x[2:]
195     data = raw.split(",")
196     #cur.execute("SET SEARCH_PATH to piraan;")
197     sql_query = f'''

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

PS C:\Users\mattb\OneDrive\Documents\Study programming\UEA\coursework\Database assessment 2\My test files\Python files\python scripts> python3 runme.py
[(datetime.date(2019, 4, 6), 'Bantarsari Kulon', 0), (datetime.date(2019, 4, 10), 'Lodon Wetan', 0), (datetime.date(2019, 4, 1), 'Trzcina', 4), (datetime.date(2019, 4, 26), 'Lapa do Lobo', 0), (datetime.date(2019, 4, 12), 'Perehoniwka', 0), (datetime.date(2019, 4, 28), 'London', 3), (datetime.date(2019, 4, 1), 'Stadium 1', 5)]
PS C:\Users\mattb\OneDrive\Documents\Study programming\UEA\coursework\Database assessment 2\My test files\Python files\python scripts> []

TASK Q:

```

1  Q:
2  |-----|
3  | ecode | edesc | amount |
4  |-----|
5  | 5200 | 400m swim | 1 |
6  | 5600 | 400m swim | 4 |
7  | 5400 | 400m swimming event | 3 |
8  | P5SL | dolor vel est donec | 4 |
9  |-----|

270 INNER JOIN spectator ON ticket.sno = spectator.sno
271 WHERE ticket.sno = {data[0]};
272
273 sql_query = f'SELECT * FROM sitinerary'
274
275 try:
276     cur.execute(sql_drop)
277     cur.execute(sql_create)
278     cur.execute(sql_insert)
279     cur.execute(sql_query)
280 except Exception as e:
281     clearOutput()
282     writeOutput(print_error(e))
283     conn.commit()
284     clearOutput()
285     rows = cur.fetchall()
286     print(rows)
287     for row in rows:
288         #print(item, ", ", end='')
289         s = str(row)
290         writeOutput(s + '\n')
291 elif(x[0] == 'T'):
292     raw = x[2:]
293     data = raw.split(",")
294     #cur.execute("SET SEARCH_PATH to piraan;")
295     sql_query = f'''SELECT tno, ecode, sno, check_ticket_in_ticket as
296     ticket_status, sname FROM all_tickets_with_info WHERE ecode = {data
297     [0]} AND check_ticket_in_ticket = false;'''
298     try:
299         cur.execute(sql_query)
300     except Exception as e:
301         clearOutput()
302         writeOutput(print_error(e))
303         conn.commit()
304         clearOutput()
305         rows = cur.fetchall()
306         for row in rows:
307             #print(item, ", ", end='')
308             s = str(row)
309             writeOutput(s + '\n')
310 elif(x[0] == 'V'):
311     raw = x[2:]
312     data = raw.split(",")

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

[(datetime.date(2019, 4, 6), 'Bantarsari Kulon', 0), (datetime.date(2019, 4, 10), 'Lodon Wetan', 0), (datetime.date(2019, 4, 1), 'Trzcina', 4), (datetime.date(2019, 4, 26), 'Lapa do Lobo', 0), (datetime.date(2019, 4, 12), 'Perehoniwka', 0), (datetime.date(2019, 4, 28), 'London', 3), (datetime.date(2019, 4, 1), 'Stadium 1', 5)]
PS C:\Users\mattb\OneDrive\Documents\Study programming\UEA\coursework\Database assessment 2\My test files\Python files\python scripts> python3 runme.py
PS C:\Users\mattb\OneDrive\Documents\Study programming\UEA\coursework\Database assessment 2\My test files\Python files\python scripts> []

TASK R:

TEST 1 PASS R S400

The screenshot shows a Visual Studio Code editor with a Python script named `runme.py` and its output in the `Output` window. The script is a Python program that interacts with a database. It has a menu with options 'X' and 'Z'. Option 'X' prints a task description and truncates several tables. Option 'Z' truncates all tables. The output window shows the result of running the program, which is 'R S400'.

```
pt.add_row([row[0],row[1],row[2],row[3],
row[4]])
writeOutput('TASK V: \n' + pt.get_string())
elif(x[0] == 'X'):
    cur.close()
    conn.close()
    clearOutput()
    writeOutput('You have exited the program!')
elif(x[0] == 'Z'):
    sql_query = f'''TRUNCATE event CASCADE;
TRUNCATE ticket CASCADE;
TRUNCATE cancel CASCADE;
TRUNCATE spectator CASCADE;
TRUNCATE sitinerary CASCADE;
'''
    try:
        cur.execute(sql_query)
    except Exception as e:
        clearOutput()
        writeOutput(print_error(e))
    conn.commit()
    clearOutput()
    writeOutput('Z. Tables cleared')
except Exception as e:
    print(e)
```

Output:

ecode	edesc	amount
S400	400m swimming event	3

Output:

```
R S400
```

TEST 2 PASS R S600

The screenshot shows a Visual Studio Code editor with a Python script named `runme.py` and its output in the `Output` window. The script is a Python program that interacts with a database. It has a menu with options 'R' and 'Z'. Option 'R' creates a view and prints its contents. Option 'Z' truncates all tables. The output window shows the result of running the program, which is 'R S600'.

```
writeOutput('Q: \n' + pt.get_string())
elif(x[0] == 'R'):
    raw = x[2:]
    data = raw.split(",")
    #cur.execute("SET SEARCH_PATH to pirate;")
    sql_query = f'''
CREATE OR REPLACE VIEW viewr AS
SELECT ticket.ecode, event.edesc, COUNT(tno) FROM ticket
LEFT JOIN event ON ticket.ecode = event.ecode
WHERE ticket.ecode = '{data[0]}'
GROUP BY ticket.ecode, event.edesc;
'''
    sql_return = '''
SELECT * FROM viewr
'''
    try:
        cur.execute(sql_query)
        cur.execute(sql_return)
    except Exception as e:
        clearOutput()
        writeOutput(print_error(e))
    conn.commit()
    clearOutput()
    rows = cur.fetchall()
    pt.field_names = ['ecode', 'edesc', 'amount']
    for row in rows:
        pt.add_row([row[0], row[1], row[2]])
    writeOutput(pt.get_string())
```

Output:

ecode	edesc	amount
S600	400m swim	4

Output:

```
R S600
```

TASK 5:

TEST S 4

The screenshot displays a Windows 10 desktop environment. The primary focus is the Visual Studio Code (VS Code) editor, which is open to a Python file named `summay.py`. The script is designed to interact with a SQLite database named `sitinerary`. It begins by creating a table with columns `sno`, `sname`, `ecode`, `elocation`, `etime`, and `edesc`. The table is populated with two records: Sarah Hodgson (sno: 4, ecode: 5400, elocation: London, etime: 10:00:00) and Sarah Hodgson (sno: 4, ecode: 5600, elocation: Stadium 1, etime: 09:00:00). The script then performs a self-join on the `sitinerary` table to find events where the location and time match. The results are printed to the console using `writeOutput` and `print_error` functions. The script also includes error handling and a commit statement.

Below the editor, a terminal window is open, showing the execution of the script. The output displays the table structure and the two records inserted into the `sitinerary` table. The terminal also shows the results of the self-join query, which returns two rows, both representing Sarah Hodgson's event at Stadium 1.

The Windows taskbar at the bottom of the screen shows the Start button, a search bar, and several pinned application icons including File Explorer, Edge, and VS Code. The system tray in the bottom right corner indicates the date and time as 12/4/2020, 5:20 PM.

TEST S 7

The screenshot displays a Windows 10 desktop environment. The primary focus is a code editor window titled "Normaly 1" showing a Python script. The script defines a table structure for a database, inserts data, and then uses a cursor to execute a query that selects ticket information based on event details. The script includes error handling and output formatting.

```
#!/usr/bin/perl

use DBI;

my $dsn = "dbi:Oracle:11g";
my $username = "scott";
my $password = "scott";

my $dbh = DBI->connect($dsn, $username, $password);

my $table = "sitarney";

my $create_table_sql = "CREATE TABLE $table (
    sno NUMBER(4) NOT NULL,
    sname VARCHAR(20) NOT NULL,
    ecode VARCHAR(20) NOT NULL,
    etime TIME(8) NOT NULL,
    edesc VARCHAR(20) NOT NULL
);";

my $insert_sql = "INSERT INTO $table (sno, sname, ecode, etime, edesc) VALUES (7, 'Dave', 'S400', '2019-04-28', 'London')";

my $select_sql = "SELECT ticket.sno, spectator.sname, event.ecode, event.edate, event.elocation, event.etime, event.edesc
FROM ticket
INNER JOIN event ON ticket.ecode = event.ecode
INNER JOIN spectator ON ticket.sno = spectator.sno
WHERE ticket.sno = {data[0]}";

my $sql_query = "SELECT * FROM sitarney";

try {
    $cur = $dbh->prepare($create_table_sql);
    $cur->execute($create_table_sql);
    $cur->execute($insert_sql);
    $cur->execute($select_sql);
} catch {
    my $e = $_;
    print "Error: $e\n";
}

my $rows = $cur->fetchall();
my $field_names = ["sno", "sname", "ecode", "elocation", "etime", "edesc"];

for my $row ($rows) {
    print join("\t", ($field_names[0], $row[0], $row[1], $row[2], $row[3], $row[4], $row[5]));
    print "\n";
}

my $row = $cur->fetchrow();
my $data = $row->split("\t");
my $sql_query = "SELECT $table.sno, $table.sname, $table.edate, $table.elocation, $table.etime, $table.edesc
FROM $table
WHERE $table.sno = {data[0]}";

my $cur = $dbh->prepare($sql_query);
$cur->execute($sql_query);
my $rows = $cur->fetchall();
my $field_names = ["sno", "sname", "ecode", "etime", "edesc"];

for my $row ($rows) {
    print join("\t", ($field_names[0], $row[0], $row[1], $row[2], $row[3], $row[4], $row[5]));
    print "\n";
}
```

The terminal window at the bottom shows the execution of the script, displaying the output of the SQL queries. The file explorer on the right shows the directory structure of the project, including a "data" folder and a "sitarney" table.

TASK T

TEST 1 : T 5

The screenshot shows the Visual Studio Code interface with the file `runme.py` open. The script contains logic to query a database and output results. The output window displays the following table:

sname	ecode	ticket_status
Anna Bagnall	5600	True

The script code includes database connection, query execution, and error handling. The terminal shows the command `python3 runme.py` being executed.

TEST 2: T 51

The screenshot shows the Visual Studio Code interface with the file `runme.py` open. The script contains logic to query a database and output results. The output window displays the following table:

sname	ecode	ticket_status
David	P54T	False

The script code is identical to the one in the first screenshot. The terminal shows the command `python3 runme.py` being executed.

TEST 3: T 52

```

1 TASK T:
2 |-----|
3 | sname | ecode | ticket_status |
4 |-----|
5 | Sarah Hodgson | S400 | False |
6 |-----|

```

```

297 pt.add_row([row[0],row[1],row[2]])
298 writeOutput('TASK S: \n' + pt.get_string()
299 elif(x[0] == 'T'):
300     raw = x[2:]
301     data = raw.split(",")
302     #cur.execute("SET SEARCH_PATH to pirean;")
303     sql_query = f'''
304     SELECT sname, ecode, check_ticket_in_ticket as
305     ticket_status FROM all_tickets_with_info WHERE tno = '
306     {data[0]}';'''
307     try:
308         cur.execute(sql_query)
309     except Exception as e:
310         clearOutput()
311         writeOutput(print_error(e))
312     conn.commit()
313     clearOutput()
314     rows = cur.fetchall()
315     pt.field_names = ['sname', 'ecode', 'ticket_status']
316     for row in rows:
317         pt.add_row([row[0],row[1],row[2]])
318     writeOutput('TASK T: \n' + pt.get_string())
319 elif(x[0] == 'V'):
320     raw = x[2:]
321     data = raw.split(",")
322     #cur.execute("SET SEARCH_PATH to pirean;")
323     sql_query = f'''SELECT * FROM all_tickets_with_info
324     WHERE ecode = '{data[0]}' AND ticket_status = true;'''
325     try:
326         cur.execute(sql_query)
327     except Exception as e:
328         clearOutput()
329         writeOutput(print_error(e))
330     conn.commit()
331     clearOutput()
332     rows = cur.fetchall()

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

PS C:\Users\matthb\OneDrive\Documents\Study programming UEA\coursework\Database assessment 2\My test files\Python files\python script> python3 runme.py

PS C:\Users\matthb\OneDrive\Documents\Study programming UEA\coursework\Database assessment 2\My test files\Python files\python script> []

Final ticket status table (multiple customers have multiple tickets so names are similar)

```

pirean=# select * from all_tickets_with_info;
tno | ecode | sno | check_ticket_in_ticket | sname
-----+-----+-----+-----+-----
2 | S200 | 2 | t | Sam Bagnall
3 | S400 | 4 | t | Sarah Hodgson
4 | S600 | 4 | t | Sarah Hodgson
5 | S600 | 3 | t | Anna Bagnall
6 | P55L | 3 | t | Anna Bagnall
7 | P55L | 5 | t | Dave
9 | S400 | 7 | t | Dave
10 | P55L | 7 | t | Dave
11 | S600 | 8 | t | David
12 | S600 | 6 | t | David
13 | P55L | 6 | t | David
20 | S400 | 3 | t | Anna Bagnall
50 | P55L | 8 | f | David
51 | P54T | 6 | f | David
52 | S400 | 4 | f | Sarah Hodgson
53 | P55L | 3 | f | Anna Bagnall
54 | P55L | 8 | f | David
(17 rows)

```

TASK V

TEST 1: S400

```
1 TASK V:
2
3 | tno | ecode | sno | ticket_status | sname |
4 |-----|-----|-----|-----|-----|
5 | 52 | S400 | 4 | False | Sarah Hodgson |
6 |-----|-----|-----|-----|
```

```
314 for row in rows:
315     pt.add_row([row[0],row[1],row[2]])
316     writeOutput('TASK T: \n' + pt.get_string())
317 elif(x[0] == 'V'):
318     raw = x[2:]
319     data = raw.split(',')
320     #cur.execute("SET SEARCH_PATH to pirean;")
321     sql_query = f'''SELECT tno, ecode, sno,
322 check_ticket_in_ticket as ticket_status,
323 sname FROM all_tickets_with_info WHERE ecode
324 = '{data[0]}' AND check_ticket_in_ticket =
325 false;'''
326 try:
327     cur.execute(sql_query)
328 except Exception as e:
329     clearOutput()
330     writeOutput(print_error(e))
331 conn.commit()
332 clearOutput()
333 rows = cur.fetchall()
334 pt.field_names = ['tno', 'ecode', 'sno',
335 'ticket_status', 'sname']
336 for row in rows:
337     pt.add_row([row[0],row[1],row[2],row[3],
338 row[4]])
339 writeOutput('TASK V: \n' + pt.get_string())
340 elif(x[0] == 'X'):
341     cur.close()
342     conn.close()
343     print("You have exited the program")
```

no results to fetch
PS C:\Users\matth\OneDrive\Documents\Study programming UEA\coursework\Database assessment 2\My test files\Python files\python script> python3 runme.py
no results to fetch
PS C:\Users\matth\OneDrive\Documents\Study programming UEA\coursework\Database assessment 2\My test files\Python files\python script> python3 runme.py
no results to fetch
PS C:\Users\matth\OneDrive\Documents\Study programming UEA\coursework\Database assessment 2\My test files\Python files\python script> python3 runme.py
no results to fetch
PS C:\Users\matth\OneDrive\Documents\Study programming UEA\coursework\Database assessment 2\My test files\Python files\python script> python3 runme.py

TEST 2 P55L

```
1 TASK V:
2
3 | tno | ecode | sno | ticket_status | sname |
4 |-----|-----|-----|-----|-----|
5 | 53 | P55L | 3 | False | Anna Bagnall |
6 | 54 | P55L | 8 | False | David |
7 | 50 | P55L | 8 | False | David |
8 |-----|-----|-----|-----|
```

```
314 for row in rows:
315     pt.add_row([row[0],row[1],row[2]])
316     writeOutput('TASK T: \n' + pt.get_string())
317 elif(x[0] == 'V'):
318     raw = x[2:]
319     data = raw.split(',')
320     #cur.execute("SET SEARCH_PATH to pirean;")
321     sql_query = f'''SELECT tno, ecode, sno,
322 check_ticket_in_ticket as ticket_status,
323 sname FROM all_tickets_with_info WHERE ecode
324 = '{data[0]}' AND check_ticket_in_ticket =
325 false;'''
326 try:
327     cur.execute(sql_query)
328 except Exception as e:
329     clearOutput()
330     writeOutput(print_error(e))
331 conn.commit()
332 clearOutput()
333 rows = cur.fetchall()
334 pt.field_names = ['tno', 'ecode', 'sno',
335 'ticket_status', 'sname']
336 for row in rows:
337     pt.add_row([row[0],row[1],row[2],row[3],
338 row[4]])
339 writeOutput('TASK V: \n' + pt.get_string())
340 elif(x[0] == 'X'):
341     cur.close()
342     conn.close()
343     print("You have exited the program")
```

no results to fetch
PS C:\Users\matth\OneDrive\Documents\Study programming UEA\coursework\Database assessment 2\My test files\Python files\python script> python3 runme.py
no results to fetch
PS C:\Users\matth\OneDrive\Documents\Study programming UEA\coursework\Database assessment 2\My test files\Python files\python script> python3 runme.py
no results to fetch
PS C:\Users\matth\OneDrive\Documents\Study programming UEA\coursework\Database assessment 2\My test files\Python files\python script> python3 runme.py
no results to fetch
PS C:\Users\matth\OneDrive\Documents\Study programming UEA\coursework\Database assessment 2\My test files\Python files\python script> python3 runme.py
no results to fetch
PS C:\Users\matth\OneDrive\Documents\Study programming UEA\coursework\Database assessment 2\My test files\Python files\python script> python3 runme.py

TASK X

The image shows a Visual Studio Code editor window with a Python file named 'runme.py' open. The script is designed to interact with a database using a cursor. It starts by splitting a raw input string into a list. Then, it constructs a SQL query to select ticket information based on an 'ecode' value. The script uses a cursor to execute this query, fetches the results, and prints them in a formatted table. It also includes error handling for database exceptions. The script ends with a 'try:' block that contains the main logic, followed by a 'finally:' block that closes the cursor and connection. The terminal at the bottom shows the output of the script, which is a table of ticket data. The terminal also shows the command 'python3 runme.py' being executed multiple times, and the output of the script is displayed for each execution. The terminal output shows a table with columns 'tno', 'ecode', 'sno', and 'ticket_status'. The first row of data is: tno: 1, ecode: 1, sno: 1, ticket_status: 1. The second row of data is: tno: 2, ecode: 2, sno: 2, ticket_status: 2. The third row of data is: tno: 3, ecode: 3, sno: 3, ticket_status: 3. The fourth row of data is: tno: 4, ecode: 4, sno: 4, ticket_status: 4. The fifth row of data is: tno: 5, ecode: 5, sno: 5, ticket_status: 5. The sixth row of data is: tno: 6, ecode: 6, sno: 6, ticket_status: 6. The seventh row of data is: tno: 7, ecode: 7, sno: 7, ticket_status: 7. The eighth row of data is: tno: 8, ecode: 8, sno: 8, ticket_status: 8. The ninth row of data is: tno: 9, ecode: 9, sno: 9, ticket_status: 9. The tenth row of data is: tno: 10, ecode: 10, sno: 10, ticket_status: 10. The terminal also shows the command 'python3 runme.py' being executed multiple times, and the output of the script is displayed for each execution. The terminal output shows a table with columns 'tno', 'ecode', 'sno', and 'ticket_status'. The first row of data is: tno: 1, ecode: 1, sno: 1, ticket_status: 1. The second row of data is: tno: 2, ecode: 2, sno: 2, ticket_status: 2. The third row of data is: tno: 3, ecode: 3, sno: 3, ticket_status: 3. The fourth row of data is: tno: 4, ecode: 4, sno: 4, ticket_status: 4. The fifth row of data is: tno: 5, ecode: 5, sno: 5, ticket_status: 5. The sixth row of data is: tno: 6, ecode: 6, sno: 6, ticket_status: 6. The seventh row of data is: tno: 7, ecode: 7, sno: 7, ticket_status: 7. The eighth row of data is: tno: 8, ecode: 8, sno: 8, ticket_status: 8. The ninth row of data is: tno: 9, ecode: 9, sno: 9, ticket_status: 9. The tenth row of data is: tno: 10, ecode: 10, sno: 10, ticket_status: 10.

TASK Z

The screenshot displays a Visual Studio Code editor with a Python script named `runme.py`. The script is designed to interact with a database, likely SQLite, using the `sqlite3` module. It features a `try-except` block to handle potential database errors. The script performs the following actions:

- Connects to a database (the connection string is not fully visible but appears to be `sqlite:///test.db`).
- Executes a `TRUNCATE` statement on a table named `event`.
- Writes a message to the output: "You have exited the program!".
- Prints the output of the `TRUNCATE` statement.

The output window at the bottom shows the execution of the script, which prints "Z. Tables cleared". The terminal also shows the command `python3 runme.py` being executed in a PowerShell environment.

```

1  pt.add_row([row[0],row[1],row[2],row[3],
2  row[4]])
3  writeOutput('TASK V: \n' + pt.get_string())
4  elif(x[0] == 'X'):
5      cur.close()
6      conn.close()
7      clearOutput()
8      writeOutput("You have exited the program!")
9  elif(x[0] == 'Z'):
10     sql_query = f'''TRUNCATE event CASCADE;
11                     TRUNCATE ticket CASCADE;
12                     TRUNCATE cancel CASCADE;
13                     TRUNCATE spectator CASCADE;
14                     TRUNCATE sitinerary CASCADE;
15                     '''
16
17     try:
18         cur.execute(sql_query)
19     except Exception as e:
20         clearOutput()
21         writeOutput(print_error(e))
22     conn.commit()
23     clearOutput()
24     writeOutput("Z. Tables cleared")
25 except Exception as e:
26     print(e)
27
28
29

```

```

t: powershell
PS C:\Users\matth\OneDrive\Documents\Study programming\UEA\coursework\Database assessment 2\My test files\Python files\python script> python3 runme.py
PS C:\Users\matth\OneDrive\Documents\Study programming\UEA\coursework\Database assessment 2\My test files\Python files\python script> python3 runme.py
PS C:\Users\matth\OneDrive\Documents\Study programming\UEA\coursework\Database assessment 2\My test files\Python files\python script> python3 runme.py
You have exited the program
PS C:\Users\matth\OneDrive\Documents\Study programming\UEA\coursework\Database assessment 2\My test files\Python files\python script> python3 runme.py
You have exited the program
PS C:\Users\matth\OneDrive\Documents\Study programming\UEA\coursework\Database assessment 2\My test files\Python files\python script> python3 runme.py
PS C:\Users\matth\OneDrive\Documents\Study programming\UEA\coursework\Database assessment 2\My test files\Python files\python script>

```