Using Functions, Subqueries, and ROLAP in SQL Queries

Physical Table Design of Sale_Co_DW.db, QC_Checks.db, or Company.db

This physical table structure for the three SQLite databases may be obtained via DBBrowser for SQLite or via .schema dot command in sqlite3.exe.

Sale_Co_DW.db is more of a data warehouse design having fact tables and dimension tables.

QC_Checks.db contains quality check errors from the case study reviewed during Week 05.

Company.db contains company data used to illustrate subqueries in Classwork 6.2.

Exercise:

1) Use any of the three databases above to complete each subsection in step 2. Do not use any of the subqueries illustrated in Classwork 6.1 or Classwork 6.2.

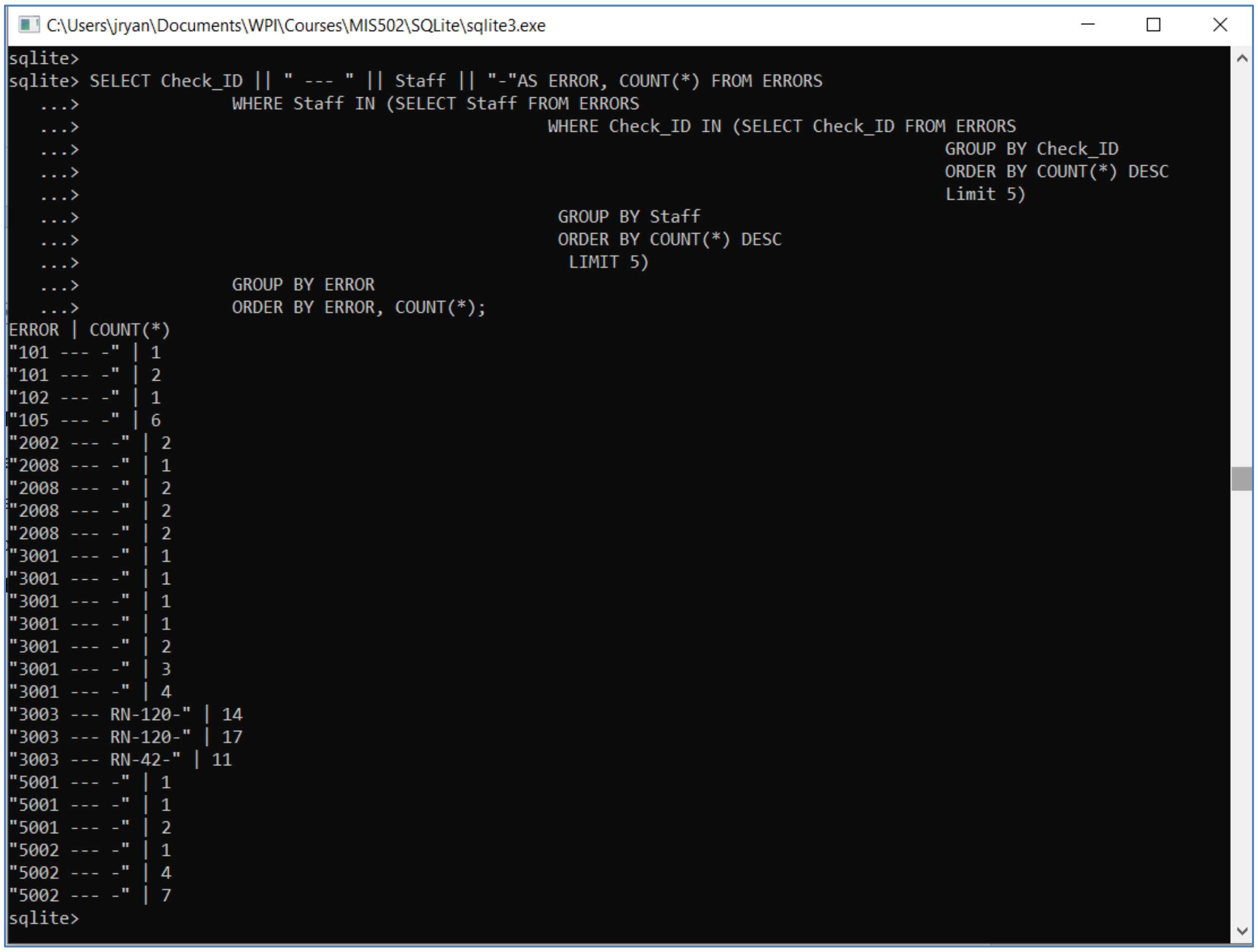
2) Create five unique, executable queries (2 points), using a minimum of four functions on average (2 points), having multiple grouping indexes (2 points), and :
   a. A Type I subquery (2 points) nested with two inner queries (2 points).
   b. A Type II subquery (2 points) nested with two inner queries (2 points).
   c. A Type III correlated subquery (2 points).
   d. The SELECT projection from a table created by a SELECT statement (2 points) with 5 columns.
   e. The SELECT projection from tables saved to a CSV file (2 points).

3) Submit to Canvas Assignment in one PDF document:
   a. Your SQL scripts for each query.
   b. Legible output projection from running each query.

Answers using QC_Checks.db:

2a Type I subquery (2 points) nested with two inner queries (2 points).

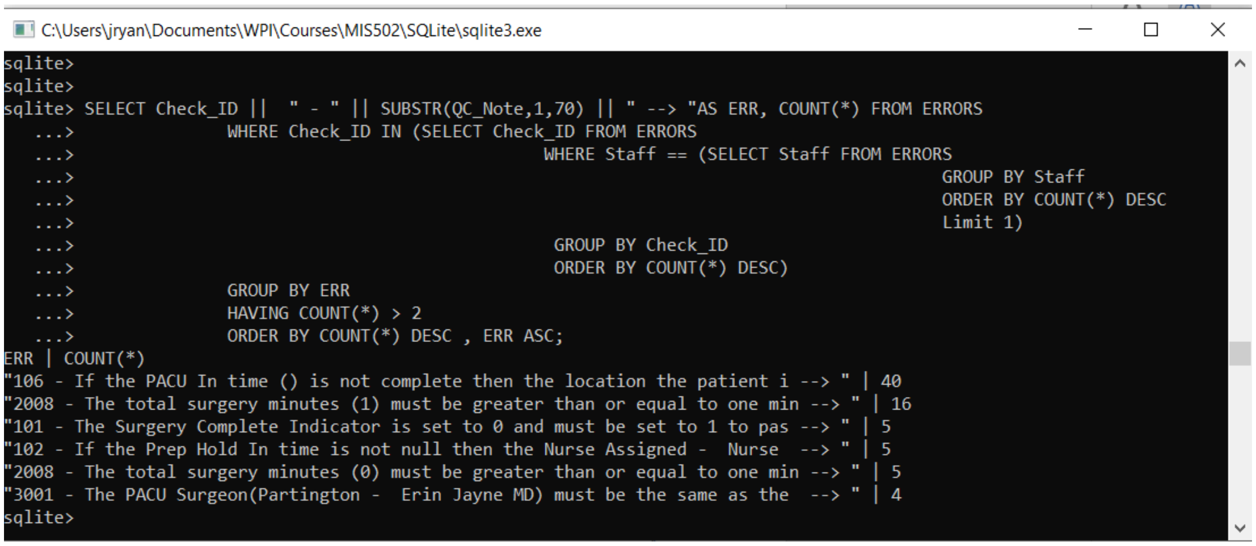
```
.open QC_Checks.db
.headers on
.separator “ | “
SELECT Check_ID || “ --- ” || Staff || “—“AS ERR, COUNT(*) FROM ERRORS
        WHERE Staff IN (SELECT Staff FROM ERRORS
                            WHERE Check_ID IN (SELECT Check_ID FROM ERRORS
                                                    GROUP BY Check_ID
                                                    ORDER BY COUNT(*) DESC
                                                    Limit 5)
                            GROUP BY Staff
                            ORDER BY COUNT(*) DESC
                             LIMIT 5)
        GROUP BY ERR
        ORDER BY ERR, COUNT(*);
```



```
sqlite>
sqlite> SELECT Check_ID || " --- " || Staff || "-"AS ERROR, COUNT(*) FROM ERRORS
   ...>                 WHERE Staff IN (SELECT Staff FROM ERRORS
   ...>                                                    WHERE Check_ID IN (SELECT Check_ID FROM ERRORS
   ...>                                                                                     GROUP BY Check_ID
   ...>                                                                                     ORDER BY COUNT(*) DESC
   ...>                                                                                     Limit 5)
   ...>                                                     GROUP BY Staff
   ...>                                                     ORDER BY COUNT(*) DESC
   ...>                                                      LIMIT 5)
   ...>                 GROUP BY ERROR
   ...>                 ORDER BY ERROR, COUNT(*);
ERROR | COUNT(*)
"101 --- -" | 1
"101 --- -" | 2
"102 --- -" | 1
"105 --- -" | 6
"2002 --- -" | 2
"2008 --- -" | 1
"2008 --- -" | 2
"2008 --- -" | 2
"2008 --- -" | 2
"3001 --- -" | 1
"3001 --- -" | 1
"3001 --- -" | 1
"3001 --- -" | 1
"3001 --- -" | 2
"3001 --- -" | 3
"3001 --- -" | 4
"3003 --- RN-120-" | 14
"3003 --- RN-120-" | 17
"3003 --- RN-42-" | 11
"5001 --- -" | 1
"5001 --- -" | 1
"5001 --- -" | 2
"5002 --- -" | 1
"5002 --- -" | 4
"5002 --- -" | 7
sqlite>
```

2b: A Type II subquery (2 points) nested with two inner queries (2 points).

```
SELECT Check_ID ||  " — " || SUBSTR(QC_Note,1,70) || " --> "AS ERR, COUNT(*) FROM ERRORS
        WHERE Check_ID IN (SELECT Check_ID FROM ERRORS
                          WHERE Staff == (SELECT Staff FROM ERRORS
                                         GROUP BY Staff
                                         ORDER BY COUNT(*) DESC
                                         Limit 1)
                          GROUP BY Check_ID
                          ORDER BY COUNT(*) DESC)
        GROUP BY ERR
        HAVING COUNT(*) > 2
        ORDER BY COUNT(*) DESC , ERR ASC;
```



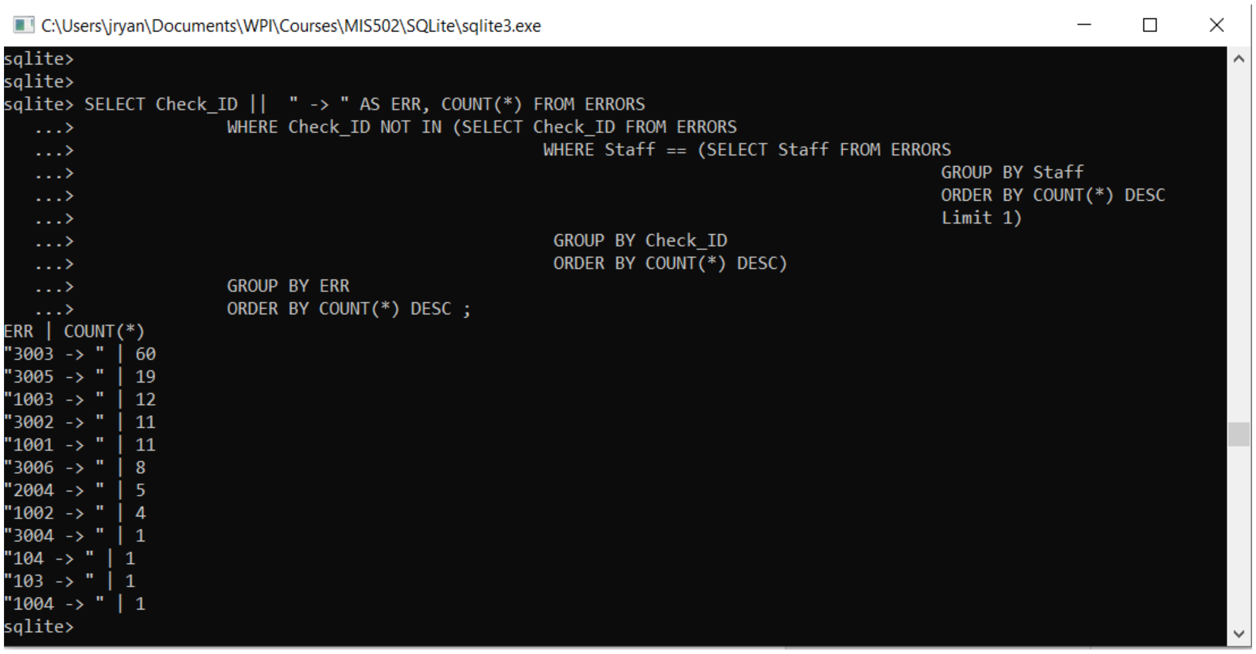
C:\Users\jryan\Documents\WPI\Courses\MIS502\SQLite\sqlite3.exe

```
sqlite>
sqlite>
sqlite> SELECT Check_ID ||  " - " || SUBSTR(QC_Note,1,70) || " --> "AS ERR, COUNT(*) FROM ERRORS
   ...>              WHERE Check_ID IN (SELECT Check_ID FROM ERRORS
   ...>                                    WHERE Staff == (SELECT Staff FROM ERRORS
   ...>                                                           GROUP BY Staff
   ...>                                                           ORDER BY COUNT(*) DESC
   ...>                                                           Limit 1)
   ...>                                     GROUP BY Check_ID
   ...>                                     ORDER BY COUNT(*) DESC)
   ...>              GROUP BY ERR
   ...>              HAVING COUNT(*) > 2
   ...>              ORDER BY COUNT(*) DESC , ERR ASC;
ERR | COUNT(*)
"106 - If the PACU In time () is not complete then the location the patient i --> " | 40
"2008 - The total surgery minutes (1) must be greater than or equal to one min --> " | 16
"101 - The Surgery Complete Indicator is set to 0 and must be set to 1 to pas --> " | 5
"102 - If the Prep Hold In time is not null then the Nurse Assigned -  Nurse  --> " | 5
"2008 - The total surgery minutes (0) must be greater than or equal to one min --> " | 5
"3001 - The PACU Surgeon(Partington -  Erin Jayne MD) must be the same as the  --> " | 4
sqlite>
```

2c: A Type III correlated subquery (2 points).

```
SELECT Check_ID ||  “ —> “ AS ERR, COUNT(*) FROM ERRORS
          WHERE Check_ID NOT IN (SELECT Check_ID FROM ERRORS
                                 WHERE Staff == (SELECT Staff FROM ERRORS
                                                 GROUP BY Staff
                                                 ORDER BY COUNT(*) DESC
                                                 Limit 1)
                                 GROUP BY Check_ID
                                 ORDER BY COUNT(*) DESC)
          GROUP BY ERR
          ORDER BY COUNT(*) DESC ;
```

The correlated subquery above is an example of using NOT IN rather than NOT EXISTS.

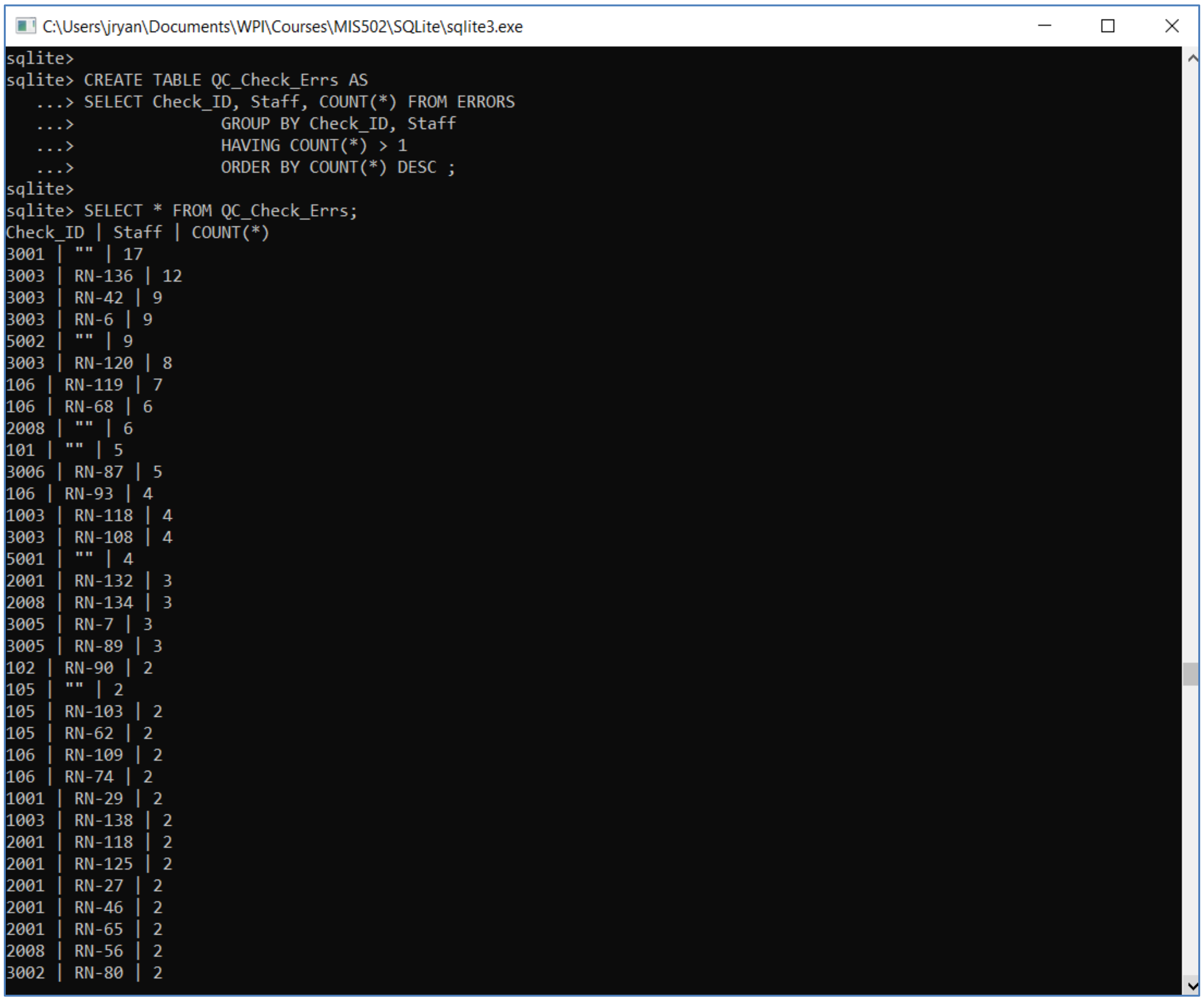


C:\Users\jryan\Documents\WPI\Courses\MIS502\SQLite\sqlite3.exe

```
sqlite>
sqlite>
sqlite> SELECT Check_ID ||  " -> " AS ERR, COUNT(*) FROM ERRORS
   ...>              WHERE Check_ID NOT IN (SELECT Check_ID FROM ERRORS
   ...>                                        WHERE Staff == (SELECT Staff FROM ERRORS
   ...>                                                                    GROUP BY Staff
   ...>                                                                    ORDER BY COUNT(*) DESC
   ...>                                                                    Limit 1)
   ...>                                         GROUP BY Check_ID
   ...>                                         ORDER BY COUNT(*) DESC)
   ...>              GROUP BY ERR
   ...>              ORDER BY COUNT(*) DESC ;
ERR | COUNT(*)
"3003 -> " | 60
"3005 -> " | 19
"1003 -> " | 12
"3002 -> " | 11
"1001 -> " | 11
"3006 -> " | 8
"2004 -> " | 5
"1002 -> " | 4
"3004 -> " | 1
"104 -> " | 1
"103 -> " | 1
"1004 -> " | 1
sqlite>
```

2d: A SELECT projection from a table created by a SELECT statement (2 points) with 5 columns.

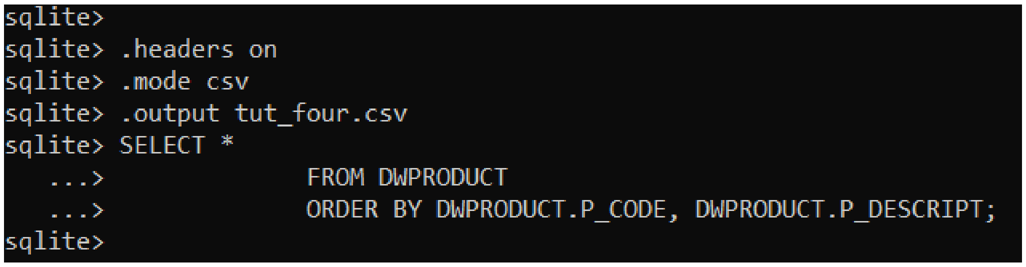
```
CREATE TABLE QC_Check_Errs AS
SELECT Check_ID, Staff, COUNT(*) FROM ERRORS
        GROUP BY Check_ID, Staff
        HAVING COUNT(*) > 1
        ORDER BY COUNT(*) DESC ;

SELECT * FROM QC_Check_Errs;
```



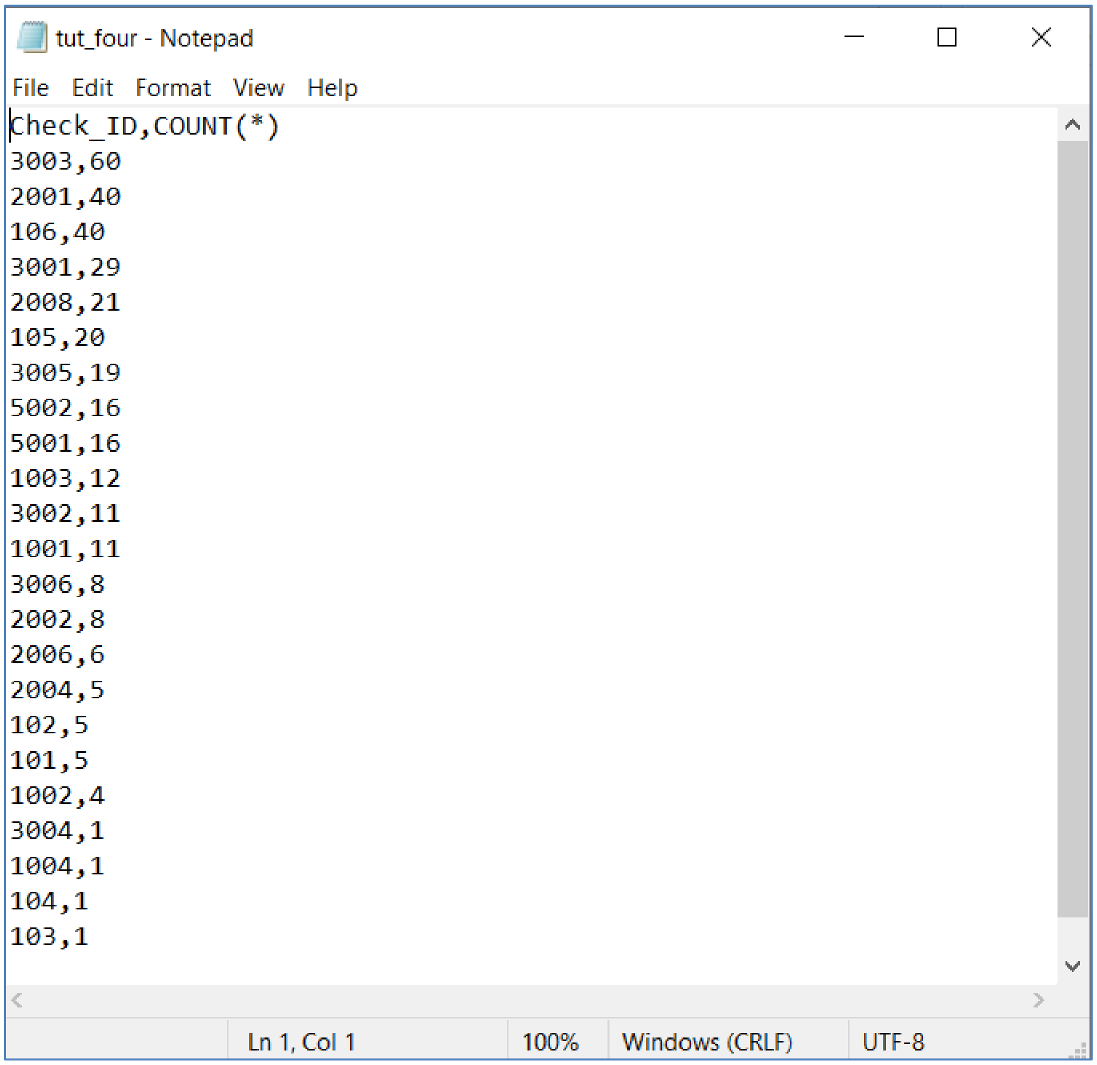
```
sqlite>
sqlite> CREATE TABLE QC_Check_Errs AS
   ...> SELECT Check_ID, Staff, COUNT(*) FROM ERRORS
   ...>               GROUP BY Check_ID, Staff
   ...>               HAVING COUNT(*) > 1
   ...>               ORDER BY COUNT(*) DESC ;
sqlite>
sqlite> SELECT * FROM QC_Check_Errs;
Check_ID | Staff | COUNT(*)
3001 | "" | 17
3003 | RN-136 | 12
3003 | RN-42 | 9
3003 | RN-6 | 9
5002 | "" | 9
3003 | RN-120 | 8
106 | RN-119 | 7
106 | RN-68 | 6
2008 | "" | 6
101 | "" | 5
3006 | RN-87 | 5
106 | RN-93 | 4
1003 | RN-118 | 4
3003 | RN-108 | 4
5001 | "" | 4
2001 | RN-132 | 3
2008 | RN-134 | 3
3005 | RN-7 | 3
3005 | RN-89 | 3
102 | RN-90 | 2
105 | "" | 2
105 | RN-103 | 2
105 | RN-62 | 2
106 | RN-109 | 2
106 | RN-74 | 2
1001 | RN-29 | 2
1003 | RN-138 | 2
2001 | RN-118 | 2
2001 | RN-125 | 2
2001 | RN-27 | 2
2001 | RN-46 | 2
2001 | RN-65 | 2
2008 | RN-56 | 2
3002 | RN-80 | 2
```

2e: The SELECT projection from tables saved to a CSV file (2 points).

```
.headers on
.mode csv
.output tut_four.csv
SELECT Check_ID, COUNT(*)
        FROM ERRORS
        GROUP BY Check_ID
        ORDER BY COUNT(*) DESC;
```



```
sqlite>
sqlite> .headers on
sqlite> .mode csv
sqlite> .output tut_four.csv
sqlite> SELECT *
   ...>                 FROM DWPRODUCT
   ...>                 ORDER BY DWPRODUCT.P_CODE, DWPRODUCT.P_DESCRIPT;
sqlite>
```



```
Check_ID,COUNT(*)
3003,60
2001,40
106,40
3001,29
2008,21
105,20
3005,19
5002,16
5001,16
1003,12
3002,11
1001,11
3006,8
2002,8
2006,6
2004,5
102,5
101,5
1002,4
3004,1
1004,1
104,1
103,1
```