---------------------------------------------------------------------------

This file contains several examples of querying single tables. Repeat these

exercises and experiment with some other SQLITE queries.

---------------------------------------------------------------------------

SQLITE> -- Students living in Oxford (\* -- == one line comment\*)

SQLITE> select NAME from STUDENT where CITY = 'Oxford';

NAME

--------

John

Anne

SQLITE> -- Female students not living in London

SQLITE> select NAME from STUDENT where SEX = 'F' and CITY <> 'London';

NAME

--------

Anne

SQLITE> -- Pre-requisites for 8027

SQLITE> select \* from PREQ where MCODE = '8027';

MCOD PCOD

---- ----

8027 8011

8027 8049

SQLITE> -- Reasonably good students on 8003

SQLITE> select \* from REGISTER where MCODE = '8003' and MARK between 60 and 75;

MCOD SNAME MARK

---- -------- ----------

8003 Mary 64

8003 Zoe 64

SQLITE> -- Show details of programming modules

SQLITE> select \* from MODULE where TITLE like '%Prog%';

CODE TITLE S CREDIT

---- --------------- - ----------

8003 Programming 1 B 1

8008 Programming 2 B 1

8012 OO Programming A 1

8016 Prog Structures A 1

8015 Systems Prog A 1

---------------------------------------------------------------------------

SQLITE> select \* from MODULE where TITLE like '%Prog%';

CODE TITLE S CREDIT

---- --------------- - ----------

8003 Programming 1 B 1

8008 Programming 2 B 1

8012 OO Programming A 1

8016 Prog Structures A 1

8015 Systems Prog A 1

SQLITE> -- proper projection is done via:

SQLITE> select distinct SNAME from REGISTER;

SNAME

--------

Anne

Brian

David

John

Mary

Zoe

6 rows selected.

SQLITE> -- ... as opposed to an answer with all duplicates

SQLITE> select SNAME from REGISTER;

SNAME

--------

John

Mary

David

Anne

Zoe

Brian

John

Mary

David

Anne

Zoe

Brian

John

Anne

Zoe

Brian

John

Mary

.....

David

Zoe

Brian

59 rows selected.

SQLITE> -- Group functions - e.g. calculating average mark per module

SQLITE> select MCODE, avg(MARK) MEAN from REGISTER group by MCODE;

MCOD MEAN

---- ------

8001 71.50

8002 66.00

8003 67.50

8008 69.50

8010 70.00

8011 61.75

8012 60.00

8015 67.50

8016 70.00

8019 68.67

8021 91.00

8024 65.17

8027 70.00

8034 78.00

8048 63.33

8049 68.33

8606 80.17

17 rows selected.

SQLITE> -- ... or per student

SQLITE> select SNAME, avg(MARK) MEAN

2 from REGISTER

3 group by SNAME;

SNAME MEAN

-------- ------

Anne 75.75

Brian 59.22

David 77.60

John 76.00

Mary 64.89

Zoe 63.08

6 rows selected.

SQLITE> -- Only some modules

SQLITE> -- .. and only those with more than 3 students

SQLITE> select MCODE, avg(MARK) MEAN from REGISTER

2 where MCODE in ('8003', '8008', '8011', '8016')

3 group by MCODE

4 having count(\*) > 3

5 order by MCODE desc; (\*asc by default\*)

MCOD MEAN

---- ------

8011 61.75

8008 69.50

8003 67.50

SQLITE> -- .. and other simple statistical functions

SQLITE> select MCODE, avg(MARK) AVG, min(MARK), max(MARK), count(MARK) from REGISTER group by MCODE;

MCOD AVG MIN(MARK) MAX(MARK) COUNT(MARK)

---- ------ ---------- ---------- -----------

8001 71.50 53 91 6

8002 66.00 58 76 4

8003 67.50 48 88 6

8008 69.50 62 78 4

8010 70.00 58 90 3

8011 61.75 48 72 4

8012 60.00 52 68 3

8015 67.50 63 72 2

8016 70.00 48 92 2

8019 68.67 48 93 3

8021 91.00 91 91 1

8024 65.17 45 86 6

8027 70.00 68 72 2

8034 78.00 78 78 1

8048 63.33 52 70 3

8049 68.33 60 82 3

8606 80.17 48 94 6

17 rows selected.

SQLITE> -- A bit of output formatting

SQLITE> column NAME heading Student|Name justify center

SQLITE> column SEX heading GENDER format A6 justify center

SQLITE> column FIELD format A5

SQLITE> column CITY heading Home|Town format a10

SQLITE> column MCODE heading Module format A6

SQLITE> -- the above --ain in force until new definition are given

SQLITE> select NAME, SEX, FIELD, CITY, MCODE, MARK

2 from STUDENT, REGISTER

3 where NAME=SNAME and MCODE='8049'

Student Home

Name GENDER FIELD Town Module Result

-------- ------ ----- ---------- ------ ------

Anne F SH Oxford 8049 82

Brian M COIN Banbury 8049 60

Zoe F SH London 8049 63

---------------------------------------------------------------------------

Try this which combines tree detection with formatting

---------------------------------------------------------------------------

A view is a useful concept (in fact is part of external schema) for controllingaccess to the data stored in the database. The view is a

dynamic structure - it does not involve making a physical copy of the data; it can be CREATEd and DROPped (destroyed) at any time.

---------------------------------------------------------------------------

SQLITE> -- creating a view

SQLITE> create view DB\_MARKS as select CODE, TITLE, CREDIT, SNAME, MARK

🡪 from MODULE, REGISTER

🡪 where CODE = MCODE

🡪 and CODE in ('8010', '8011', '8049', '8027');

SQLITE> select \* from DB\_MARKS;

CODE TITLE CREDIT SNAME MARK

---- --------------- ---------- -------- ----------

8027 DB Software 1 Anne 68

8027 DB Software 1 Zoe 72

8027 DB Software 1 Zoe 72

8049 Database 2 Anne 82

8049 Database 2 Brian 60

8049 Database 2 Zoe 63

8011 Data Structures 1 Anne 72

8011 Data Structures 1 John 67

8011 Data Structures 1 Mary 48

8011 Data Structures 1 Zoe 60

8010 Software Design 1 Anne 90

8010 Software Design 1 John 58

8010 Software Design 1 Zoe 62

12 rows selected.

--------------------------------------------------------------------------