# Spring 2022 courses

## DONE db Test 2

### DONE SETTINGS

Database Theory & Applications (CSC 330) test 2:

- 10 questions from quiz 4-6 (some questions modified)
- 10 new questions
- Partial credit enabled where applicable
- You can change answers before submitting
- You can resume an incomplete submission
- It is mentioned if > 1 answer is correct

Good luck!

### Joining tables

We want to `JOIN` two tables `Bought` and `Sold` on the column `product`, which exists in both tables. We want to see all columns of both tables. Which command will do the job?

TRUE:

- SELECT * FROM Bought JOIN Sold ON Bought.product = Sold.product;

FALSE:

- SELECT * FROM Bought JOIN Sold ON product;
- JOIN Bought.product ON Sold.product;
- SELECT product FROM Bought JOIN Sold;

### Print a range of values

We wish to print a range of values between `10.0` and `50.0` from a column `price` in the table `product`. Complete `[1]` and `[2]` in the statement below.

```
SELECT price FROM product WHERE product.[1] [2] 10.0 AND 50.0;
```

TRUE:

- [1] price [2] BETWEEN

FALSE:

- [1] column [2] IN
- [1] price [2] LIKE
- [1] product [2] ==

### Filter after `GROUP BY`

The `SELECT` command displays the columns `country` and `count` from a table of `companies`, and then groups by `country`. The result is filtered for the condition `COUNT(country) = 1`.

Output:

| country | count |
|---|---|
| Canada | 1 |
| China | 1 |

Which command fragment achieves this filtering after a `GROUP BY` command?

TRUE:

- HAVING COUNT(country) = 1

FALSE:

- WHERE COUNT(country) = 1
- FROM COUNT(country) = 1
- LIMIT COUNT(country) = 1

## Complete the code to return the output

Fill in the `???` in the command with the right expression to get the output shown below.

```
SELECT name
FROM companies
WHERE name ???
LIMIT 5;
```

OUTPUT:

| name |
|---|
| Oberlo |
| Root |
| Delta |
| Wire |
| Oasis |

TRUE:

- NOT LIKE 'S%'

FALSE:

- = 'S%'
- LIKE 'S_'
- LIKE 'S%'

## Aggregate functions

You can compute aggregate functions (like `AVG` or `SUM`) after the SELECT command, alongside column the column names, before the FROM command, e.g. like: `SELECT AVG(column) AS mean, column FROM table;`

TRUE

## The following command orders the table `customer` alphabetically (A-Z) by the values of the text column `name`

```
SELECT * FROM customer ORDER BY customer.name DESC;
```

FALSE

## Complete the code to return the output

Fill in `[1]` and `[2]` to get the output shown below.

```
SELECT title, country
FROM films
[1]  country = [2]
LIMIT 4;
```

OUTPUT:

| title | country |
|---|---|
| The last Emperor | China |
| Hero | China |
| House of Flying Daggers | China |
| The Promise | China |

TRUE:

- [1] WHERE [2] 'China'

FALSE:

- [1] WHEN [2] 'China'
- [1] IF [2] 'China'
- [1] WHERE [2] 'CHINA'

## Complete the code to return the output

Fill in `???` to get the output shown below.

```
SELECT ??? country
FROM films
LIMIT 5;
```

OUTPUT:

| country |
|---|
| Thailand |
| Cambodia |
| Libya |
| Turkey |

TRUE:

- DISTINCT

FALSE:

- ONLY
- LIMIT
- UNIQUE

### How would you count all rows in the `people` table?

Columns in `people` include: `name`, and `ID`. Fill the blank `???` with the correct command to return the output shown below.

*More than one answer is correct.*

```
SELECT ???
FROM people;
```

| count |
|---|
| 8397 |

TRUE:

- COUNT(*)
- count(name)

FALSE:

- NUMBER(ID)
- COUNTER(*)

### How does `ORDER BY` sort a column of text values by default?

TRUE:

- Alphabetically (A-Z)

FALSE:

- There's no natural ordering to text data
- By number of characters (fewest to most)
- Reverse alphabetically (Z-A)

### `DONE` 6 How can you find out the names of all columns in an SQLite table t if the header is off?

Tip: if the header is off, the SQLite variable `.header` is `OFF`. You can e.g. see this with the `.show` command.

TRUE:

- `.schema`

FALSE:

- `.show names`
- `SELECT * FROM t;`
- `.mode column`

Feedback: .schema contains all data definition commands (like CREATE). .show does not have an argument. SELECT would show the names IF the header was ON. .mode only changes the table layout.

## DONE 6 CHANGE What does the SQL `JOIN` operation in SQL do?

TRUE:

- It projects columns from multiple tables for a querying process

FALSE:

- It groups results by one or more columns
- It orders row output by one or more columns
- It filters rows from a table to be queried further

Feedback: The grouping command is GROUP BY. Ordering is done with ORDER BY. Filtering rows is done with WHERE, not JOIN.

## DONE 6 What's wrong with the code?

You meant to print the four lines with `id` 2-5 but the output gives you what you see below. How can you fix the command?

```
SELECT * FROM foods LIMIT 4,1;
```

```
id  type_id  name
--  -------  -----------------------
5   1        Black and White cookies
```

TRUE:

- It should be `LIMIT 1,4`

FALSE:

- It should be `LIMIT 1 OFFSET 4`
- It should be `SELECT type_id` instead of `*`
- You need a `WHERE` statement to filter the rows first

## DONE 6 Complete the SQL code to return the output

The database `people` has columns `birthdate` and `deathdate` that contain dates of birth and dates of death.

Select `[1] [2]` to get the percentage of people in `people` who are no longer alive.

CODE:

```
SELECT [1] * 100.0 / [2] AS percentage_dead FROM people;
```

OUTPUT:

```
percentage_dead
---------------
9.372394
```

TRUE:

- [1] COUNT(deathdate) [2] COUNT(*)

  FALSE:

  - [1] COUNT(*) [2] COUNT(deathdate)
  - [1] TOTAL(deathdate) [2] COUNT(*)
  - [1] COUNT(birthdate) [2] TOTAL(deathdate)

## DONE 5 Complete the code to return the output.

Get the number of German language films.

```
SELECT [1]
FROM films
WHERE language [2] [3]
```

Output:

| count |
|---:|
| 44 |

TRUE:

- `[1] COUNT(*) AS count [2] = [3] 'German'`
- `[1] COUNT(*) AS count [2] == [3] 'German'`

  FALSE:

  - `[1] count [2] = [3] 'german'`
  - `[1] COUNT(*) AS count [2] IS [3] 'German'`

  Feedback: In DataCamp, you use PostgreSQL, where `COUNT(*)` is printed as `count` without the `AS` (unlike in SQLite). - `SELECT count` queries the column `count`, and does not count. Also, we're looking for `'German'`, not `'german'` - values of string variables are case sensitive. - IS is not an operator.

  ```
  CREATE TABLE t (id INT, name TEXT);
  INSERT INTO t VALUES (1,"joe");
  INSERT INTO t VALUES (2,"jane");
  SELECT * FROM t;
  ```

  ```
  SELECT * FROM t WHERE name='joe';
  ```

## DONE 5 On the SQLite shell, SQLite commands must be entered in lowercase letters

TRUE

Feedback: by convention, yes - at least on the console, and also not all options - e.g. both `.header ON` and `.header on` work. This is just a rule. The reason is perhaps to keep SQLite as small as possible - too many choices not directly related to the data keeping would bloat the software.

## DONE 5 SQLite commands always begin with a dot (.) when you're on the SQLite shell.

Tip: the SQLite shell is where you are when you see this prompt: `sqlite>` , after starting SQLite with the command `sqlite3`.

TRUE

Feedback: E.g. .database, .tables, .show, and many other commands on the shell. When you are NOT on the shell, many SQLite commands can be executed as options for the sqlite3 command, and then they don't begin with a dot but with a dash, like in "sqlite3 -header -column" which is equivalent to .header ON and .mode column on the SQLite shell.

## DONE 4 CHANGE Which code is missing in `???` to save your database to a file `data.sql`?

Tip: the last line redirects the SQLite shell output to `stdout` (the screen).

```
sqlite> .output ???
sqlite> .dump
sqlite> .output stdout
```

TRUE:

- `data.sql`

  FALSE:

  - `sqlite.db`
  - `ON`
  - `stdin`

  Feedback: To redirect to any file, you use .output filename **-** if it does not exist, it will be created.

## DONE 4 CHANGE How can you check your table display values in SQLite?

*More than one answer correct.*

TRUE:

- With the SQL command `SELECT * FROM tbl;` (if the table `tbl` exists)
- With the SQLite command `.show`

FALSE:

- With `SELECT show;`
- With the SQLite command `.display`

Feedback: SELECT is an SQL command to query relational (table) information only. SQLite commands start with a dot and the two commands are never mixed. There is no .display command.

## DONE 4 Be the SQLite shell!

You have created a table named `newTable`, put some values into it, and saved it in a database file `data.db`. Now you want to look at the data.

Which mistakes can you identify in the following command sequence?

```
sqlite> .open data.db;
sqlite> .databases
sqlite> .tables
sqlite> SELECT * FROM newTable
```

*More than one answer is correct.*

TRUE:

- There should not be a semi-colon after the `.open` statement

- There is a semi-colon missing at the end of the `SELECT` statement

  FALSE:

  - It should be `.database`, not `.databases`
  - `.tables` needs to be followed by a table name

  Feedback: Distinguish SQLite (start with a dot) from SQL commands (end with a semi-colon). `.database` and `.tables` are both OK.

Author: Marcus Birkenkrahe

Created: 2022-03-17 Thu 15:13

Validate