

DETAILED INFORMATION OF SQL ASSESSMENT. -By: B.PRAMEELA

--> Question 2.1 What are the Top 25 schools (.edu domains)?

Ans:

```
SELECT
email_domain,
COUNT(*)
FROM users
WHERE email_domain LIKE '%.edu'
GROUP BY 1
ORDER BY 2 DESC
LIMIT 25;
```

The screenshot shows a SQL assessment interface. On the left, there's a 'Tasks' panel with a list of questions. The main area is divided into a query editor on the left and a 'Query Results' table on the right. The query editor contains the SQL code for Question 2.1. The 'Query Results' table shows the top 25 schools by email domain count.

email_domain	COUNT(*)
ucsd.edu	193
ups.edu	88
ku.edu	80
vt.edu	40
orange.state.edu	38
msu.edu	33
stethmore.edu	29
umich.edu	26
u.northwestern.edu	25
nyu.edu	23
pace.edu	22
asu.edu	22
berkeley.edu	18
ucdavis.edu	17
hawaii.neu.edu	16
chennai.edu	16
hawaii.edu	16
guelph.edu	15
utexas.edu	13
ucsc.edu	13
monmouth.edu	13
casa.edu	13
asu.edu	12
usc.edu	12
bernu.edu	12

Below the query results, there's a 'Database Schema' section showing the structure of the 'users' table:

name	type
user_id	INTEGER
learn_app	TEXT
learn_app	TEXT
learn_html	TEXT
learn_javascript	TEXT
learn_java	TEXT

-->Question 2.2 How many .edu learners are located in New York?

Ans:

```
SELECT
COUNT(*) as ny_students
FROM users
WHERE email_domain LIKE '%.edu' AND city = 'New York';
```

DETAILED INFORMATION OF SQL ASSESSMENT. -By: B.PRAMEELA

The screenshot shows the Codecademy SQL assessment interface. On the left, there's a 'Tasks' panel with a list of questions. The main area is a SQL query editor with a dark theme. The query is for Question 2.3, which asks for the count of learners using the mobile app. The query is as follows:

```
-- Question 2.3
-- SELECT
-- COUNT(*) as mobile_count
-- FROM users
-- WHERE mobile_app IS NOT NULL;
```

On the right, the 'Query Results' panel shows the database schema for the 'ny_students' database. The schema includes tables 'ny_students', 'progress', 'users', and 'mobile_count'. The 'users' table has columns: user_id (INTEGER), email_domain (TEXT), country (TEXT), city (TEXT), postal (INTEGER), mobile_app (TEXT), and sign_up_at (DATETIME). The 'mobile_count' table has a single column: count (INTEGER).

-->Question 2.3 The mobile_app column contains either mobile-user or NULL. How many of these Codecademy learners are using the mobile app?

Ans:

```
SELECT
COUNT(*) as mobile_count
FROM users
WHERE mobile_app IS NOT NULL;
```

The screenshot shows the Codecademy SQL assessment interface. On the left, there's a 'Tasks' panel with a list of questions. The main area is a SQL query editor with a dark theme. The query is for Question 3, which asks for the count of learners using the mobile app. The query is as follows:

```
-- Question 3
-- SELECT
-- COUNT(*) as mobile_count
-- FROM users
-- WHERE mobile_app IS NOT NULL;
```

On the right, the 'Query Results' panel shows the database schema for the 'ny_students' database. The schema includes tables 'ny_students', 'progress', 'users', and 'mobile_count'. The 'users' table has columns: user_id (INTEGER), email_domain (TEXT), country (TEXT), city (TEXT), postal (INTEGER), mobile_app (TEXT), and sign_up_at (DATETIME). The 'mobile_count' table has a single column: count (INTEGER).

-->Question 3

The data type of the sign_up_at column is DATETIME. It is for storing a date/time value in the database.

DETAILED INFORMATION OF SQL ASSESSMENT. -By: B.PRAMEELA

Notice that the values are formatted like:

2015-01-01 18:33:52

So the format is:

YYYY-MM-DD HH:MM:SS

SQLite comes with a strftime() function - a very powerful function that allows you to return a formatted date.

It takes two arguments:strftime(format, column)

Let's test this function out:

```
SELECT sign_up_at,
```

```
    strftime('%S', sign_up_at)
```

```
FROM users
```

```
GROUP BY 1
```

```
LIMIT 20;Now, using this function, query for the sign up counts for each hour.
```

Ans:

```
SELECT sign_up_at,
```

```
    strftime('%H', sign_up_at) as 'Hour'
```

```
FROM users
```

```
GROUP BY 1
```

```
LIMIT 20;
```

The screenshot displays a SQL assessment interface. On the left, a sidebar contains a 'Tasks' section with a list of questions and a 'Solution' section. The main area is divided into three panes: a query editor, a query results pane, and a database schema pane. The query editor shows a SQL query that uses the strftime function to format the sign_up_at column by hour. The query results pane displays the output of the query, showing the sign_up_at column and the Hour column. The database schema pane shows the structure of the users and progress tables.

sign_up_at	Hour
2017-01-02 20:27:07	20
2017-01-05 00:39:58	00
2017-01-05 03:42:07	03
2017-01-24 19:14:08	19
2017-01-24 21:58:59	21
2017-01-30 15:47:43	15
2017-02-02 18:52:50	18
2017-02-08 21:07:41	21
2017-02-10 21:26:36	21
2017-02-24 00:22:39	00
2017-03-01 05:04:14	05
2017-03-14 17:38:50	17
2017-03-16 18:15:25	18
2017-03-20 04:09:11	04
2017-03-21 17:14:59	17
2017-03-22 15:12:44	15
2017-03-24 16:20:25	16
2017-04-05 22:08:50	22
2017-04-08 19:19:17	19
2017-04-13 18:27:05	18

name	type
user_id	INTEGER
learn_cpp	TEXT
learn_sql	TEXT
learn_html	TEXT
learn_javascript	TEXT
learn_java	TEXT

Rows: 2000

users

-->Question 4 Join the two tables using JOIN and then see what you can dig out of the data!

Ans:

DETAILED INFORMATION OF SQL ASSESSMENT. -By: B.PRAMEELA

-->Update empty spaces to null values

UPDATE progress

SET learn_cpp = NULL

WHERE learn_cpp = '';

UPDATE progress

SET learn_sql = NULL

WHERE learn_sql = '';

UPDATE progress

SET learn_html = NULL

WHERE learn_html = '';

UPDATE progress

SET learn_javascript = NULL

WHERE learn_javascript = '';

UPDATE progress

SET learn_java = NULL

WHERE learn_java = '';

-->Question 4.1 Do different schools (.edu domains) prefer different courses?

Ans:

SELECT

email_domain,

ROUND(1.0 * count(p.learn_cpp)/(count(p.learn_cpp) + count(p.learn_sql) + count(p.learn_html) + count(p.learn_javascript) + count(p.learn_java)), 2) AS '%cpp',

ROUND(1.0 * COUNT(p.learn_sql)/(count(p.learn_cpp) + count(p.learn_sql) + count(p.learn_html) + count(p.learn_javascript) + count(p.learn_java)), 2) AS '%sql',

ROUND(1.0 * COUNT(p.learn_html)/(count(p.learn_cpp) + count(p.learn_sql) + count(p.learn_html) + count(p.learn_javascript) + count(p.learn_java)), 2) AS '%html',

DETAILED INFORMATION OF SQL ASSESSMENT. -By: B.PRAMEELA

ROUND(1.0 * COUNT(p.learn_javascript)/(count(p.learn_cpp) + count(p.learn_sql) + count(p.learn_html) + count(p.learn_javascript) + count(p.learn_java)), 2) AS '%javascript',

ROUND(1.0 * COUNT(p.learn_java)/(count(p.learn_cpp) + count(p.learn_sql) + count(p.learn_html) + count(p.learn_javascript) + count(p.learn_java)), 2) AS '%java'

FROM users u

JOIN progress p

ON u.user_id = p.user_id

WHERE email_domain LIKE '%.edu'

GROUP BY 1

ORDER BY (count(p.learn_cpp) + count(p.learn_sql) + count(p.learn_html) + count(p.learn_javascript) + count(p.learn_java)) DESC

LIMIT 10;

The screenshot shows a SQL assessment interface. On the left, there's a 'Tasks' panel with a list of questions. The main area is divided into a query editor on the left and a query results table on the right. The query editor contains a complex SQL query. The query results table shows the output of the query, with columns for email_domain, %cpp, %sql, %html, %javascript, and %java. The database schema diagram is also visible on the right side of the interface.

email_domain	%cpp	%sql	%html	%javascript	%java
ucledu	0.18	0.29	0.15	0.27	0.11
cp.edu	0.22	0.27	0.15	0.24	0.12
ku.edu	0.24	0.27	0.13	0.27	0.09
oregonstate.edu	0.23	0.23	0.17	0.28	0.07
vt.edu	0.28	0.2	0.18	0.28	0.08
strathmore.edu	0.2	0.3	0.19	0.22	0.11
msu.edu	0.13	0.3	0.19	0.3	0.09
myu.edu	0.2	0.27	0.13	0.31	0.09
umich.edu	0.19	0.26	0.14	0.3	0.12
pace.edu	0.2	0.3	0.15	0.3	0.05

name	type
user_id	INTEGER
learn_cpp	TEXT
learn_sql	TEXT
learn_html	TEXT
learn_javascript	TEXT
learn_java	TEXT

name	type
user_id	INTEGER
email_domain	TEXT
country	TEXT
city	TEXT
postal	INTEGER
mobile_app	TEXT
sign_up_at	DATETIME

--Question 4.2 What courses are the New Yorkers students taking?

Ans:

SELECT

ROUND(1.0 * count(p.learn_cpp)/(count(p.learn_cpp) + count(p.learn_sql) + count(p.learn_html) + count(p.learn_javascript) + count(p.learn_java)), 2) AS '%cpp',

ROUND(1.0 * COUNT(p.learn_sql)/(count(p.learn_cpp) + count(p.learn_sql) + count(p.learn_html) + count(p.learn_javascript) + count(p.learn_java)), 2) AS '%sql',

ROUND(1.0 * COUNT(p.learn_html)/(count(p.learn_cpp) + count(p.learn_sql) + count(p.learn_html) + count(p.learn_javascript) + count(p.learn_java)), 2) AS '%html',

ROUND(1.0 * COUNT(p.learn_javascript)/(count(p.learn_cpp) + count(p.learn_sql) + count(p.learn_html) + count(p.learn_javascript) + count(p.learn_java)), 2) AS '%javascript',

DETAILED INFORMATION OF SQL ASSESSMENT. -By: B.PRAMEELA

ROUND(1.0 * COUNT(p.learn_java)/(count(p.learn_cpp) + count(p.learn_sql) +
count(p.learn_html) + count(p.learn_javascript) + count(p.learn_java)), 2) AS '%java'

FROM users u

JOIN progress p

ON u.user_id = p.user_id

WHERE city = 'New York'

ORDER BY (count(p.learn_cpp) + count(p.learn_sql) + count(p.learn_html) +
count(p.learn_javascript) + count(p.learn_java)) DESC;

The screenshot shows a SQL assessment interface. On the left, there's a 'Tasks' panel with a hint: 'Should you do the inner or the outer join? Check the [List of SQL Commas](#)'. The main area is a query editor with a SQL query. Below the editor is a 'Solution:' section. On the right, there's a 'Query Results' panel showing the results of the query. The results are divided into two sections: 'Database Schema' and 'users'. The 'Database Schema' section shows a table named 'progress' with columns: name, user_id, learn_cpp, learn_html, learn_sql, learn_javascript, and learn_java. The 'users' section shows a table named 'users' with columns: name, user_id, email_domain, country, city, postal, mobile_app, and sign_up_at. The results are sorted by the sum of counts for all learning languages in descending order.

name	user_id	learn_cpp	learn_html	learn_sql	learn_javascript	learn_java
Rows: 2000						

name	user_id	email_domain	country	city	postal	mobile_app	sign_up_at
Rows: 2000							

-->Question 4.3 What courses are the Chicago students taking?

Ans:

SELECT

ROUND(1.0 * count(p.learn_cpp)/(count(p.learn_cpp) + count(p.learn_sql) + count(p.learn_html) +
count(p.learn_javascript) + count(p.learn_java)), 2) AS '%cpp',

ROUND(1.0 * COUNT(p.learn_sql)/(count(p.learn_cpp) + count(p.learn_sql) + count(p.learn_html) +
count(p.learn_javascript) + count(p.learn_java)), 2) AS '%sql',

ROUND(1.0 * COUNT(p.learn_html)/(count(p.learn_cpp) + count(p.learn_sql) + count(p.learn_html) +
count(p.learn_javascript) + count(p.learn_java)), 2) AS '%html',

ROUND(1.0 * COUNT(p.learn_javascript)/(count(p.learn_cpp) + count(p.learn_sql) +
count(p.learn_html) + count(p.learn_javascript) + count(p.learn_java)), 2) AS '%javasript',

ROUND(1.0 * COUNT(p.learn_java)/(count(p.learn_cpp) + count(p.learn_sql) + count(p.learn_html) +
count(p.learn_javascript) + count(p.learn_java)), 2) AS '%java'

FROM users u

JOIN progress p

ON u.user_id = p.user_id

DETAILED INFORMATION OF SQL ASSESSMENT. -By: B.PRAMEELA

WHERE city = 'New York'

ORDER BY (count(p.learn_cpp) + count(p.learn_sql) + count(p.learn_html) + count(p.learn_javascript) + count(p.learn_java)) DESC;

My Home

Tasks

2/5 Complete

test.sqlite

```
83 SELECT
84 ROUND(1.0 * count(p.learn_cpp)/(count(p.
85 learn_cpp) + count(p.learn_sql) + count(p.
86 learn_html) + count(p.learn_javascript) + count
87 (p.learn_java)), 2) AS "Ccpp",
88 ROUND(1.0 * COUNT(p.learn_sql)/(count(p.
89 learn_cpp) + count(p.learn_sql) + count(p.
90 learn_html) + count(p.learn_javascript) + count
91 (p.learn_java)), 2) AS "Sql",
92 ROUND(1.0 * COUNT(p.learn_html)/(count(p.
93 learn_cpp) + count(p.learn_sql) + count(p.
94 learn_html) + count(p.learn_javascript) + count
95 (p.learn_java)), 2) AS "Html",
96 ROUND(1.0 * COUNT(p.learn_javascript)/(count(p.
97 learn_cpp) + count(p.learn_sql) + count(p.
98 learn_html) + count(p.learn_javascript) + count
99 (p.learn_java)), 2) AS "Jjavascript",
100 ROUND(1.0 * COUNT(p.learn_java)/(count(p.
101 learn_cpp) + count(p.learn_sql) + count(p.
102 learn_html) + count(p.learn_javascript) + count
103 (p.learn_java)), 2) AS "Jjava"
104 FROM users u
105 JOIN progress p
106 ON u.user_id = p.user_id
107 WHERE city = 'New York'
108 ORDER BY (count(p.learn_cpp) + count(p.
109 learn_sql) + count(p.learn_html) + count(p.
110 learn_javascript) + count(p.learn_java)) DESC;
```

Save

📄

🔄

Query Results

%cpp	%sql	%html	%javascript	%java
0.2	0.2	0.2	0.2	0.2

Database Schema

name	type
user_id	INTEGER
learn_cpp	TEXT
learn_sql	TEXT
learn_html	TEXT
learn_javascript	TEXT
learn_java	TEXT

Rows: 2000

name	type
user_id	INTEGER
email_domain	TEXT
country	TEXT
city	TEXT
postal	INTEGER
mobile_app	TEXT
sign_up_at	DATETIME

Rows: 2000

2/5 Complete

Back

Finish

using JOIN and see what you get out of the data!

- Do different schools (Let domain) pi different courses?
- What courses are the New Yorkers studying?
- What courses are the Chinese students taking?

Hint

Should you do the inner or the outer join? Check the [List of SQL Commands](#)

Solution: