

```
{{ if response.serve_ad and settings.adsenseid: }} {{ pass }}  
{{ if settings.num_banners > 0 and settings.show_rs_banner: }}
```

Please Support Runestone

(/runestone/default/donate?banner={{=banner_num}})

```
{{ pass }}
```



How to query data with SQL?

Welcome! We have learned how databases work and why you might want to use them to organize your data. In this lab, we will go over *four* common types of SQL queries: Creating new records, Updating existing records, Reading records, and Deleting records (known as CRUD operations).

As a reminder, SQL queries are run on **tables**. Tables are like spreadsheets, with rows and columns. Each row represents a **record**, and each column represents a **field** in that record. For example, each **record** might be a student, with **fields** being IDs, names, grades.

student_id	name	grade
001	James Smith	A
002	Michael Brown	B+
005	Mary Wilson	A
006	Jennifer Garcia	A-
007	Elizabeth Martinez	B+
008	Patricia Anderson	A
...
015	David Taylor	A-

Tutorial Structure

This tutorial is organized around common *patterns* in SQL queries, also known as programming plans.

Each programming plan has a clear goal describing what it helps you achieve, a code template that you can modify for your own use case, and annotations on which areas to change.

Throughout the tutorial, you will first see real-world examples of how SQL is used, and then you will see how these examples break down into plans.



(start_here.html)

Getting Started

Look for these golden boxes for instructions. You will see a magnifying glass (🔍) when you need to go back to your worksheet. To start, click on the arrow on the bottom right (or the link below that says "Before You Start The Activity").

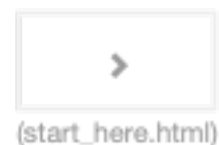
- Before You Start The Activity (start_here.html)
- Example 1: Dropping students with failing grades from the class (integrated_3.html)
 - Plan 1: Remove Records (remove_records.html)
 - Plan 2: View Records (view_records.html)
- Example 2: Awarding Grammy's and finding recent Grammy nominee songs (integrated_2.html)
 - Plan 3: Update Records Conditionally (update_records_conditionally.html)
 - Plan 4: Order Records (order_records.html)
- Example 3: Tracking new movie releases (integrated_4.html)
 - Plan 5: Add New Record (add_new_record.html)
 - Plan 6: Summarize Records (summarize_records.html)
- Exercise 1: Writing Queries for Ad Management (codewriting-1.html)
- Exercise 2: Managing Spotify Databases (codereading-1.html)
- Before You Complete The Activity - 1 (end_here_comparison.html)
- Before You Complete The Activity - 2 (end_here_usability.html)

You have attempted of activities on this page

© Copyright 2013-2020 Runestone Interactive LLC. Created using Runestone (http://runestoneinteractive.org/) 5.3.0.

| Back to

```
{{ if request.application == 'runestone': }} {{ pass }}
```



{{ if response.serve_ad and settings.adsenseid: }} {{ pass }}
{{ if settings.num_banners > 0 and settings.show_rs_banner: }}

Please Support Runestone

(/runestone/default/donate?banner={{=banner_num}})

{{ pass }}



Before You Start The Activity

Please rate how much you agree with the each statement. Your responses will be anonymized.

I am interested in learning how to manage data and run SQL queries on databases.

- 1. Strongly Disagree
- 2. Disagree
- 3. Somewhat Disagree
- 4. Neither Agree nor Disagree
- 5. Somewhat Agree
- 6. Agree
- 7. Strongly Agree

I am confident in my ability to write or edit SQL queries to analyze datasets.

- 1. Strongly Disagree
- 2. Disagree
- 3. Somewhat Disagree
- 4. Neither Agree nor Disagree
- 5. Somewhat Agree
- 6. Agree
- 7. Strongly Agree

If you completed all the activities on this page, click on the arrow on the bottom right to continue.

You have attempted of activities on this page

© Copyright 2013-2020 Runestone Interactive LLC. Created using Runestone (http://runestoneinteractive.org/) 5.3.0.

| Back to

{{ if request.application == 'runestone': }} {{ pass }}



(index.html)



(integrated_3.html)

```
{{ if response.serve_ad and settings.adsenseid: }} {{ pass }}  
{{ if settings.num_banners > 0 and settings.show_rs_banner: }}
```

Please Support Runestone

(/runestone/default/donate?banner={{=banner_num}})

```
{{ pass }}
```



Example 1: Dropping students with failing grades from the class

At the University of Illinois Urbana-Champaign, the registrar's office manages a database that includes a table called *student_grades*. This table contains records of students' grades for various courses. The university needs to ensure that records of students who are currently failing a course (with a grade below 60) are reviewed and potentially removed for academic advising purposes. Additionally, they want to review the list of students and their grades for reporting.

The *student_grades* table may include columns such as:

student_id	name	grade
001	James Smith	84
002	Michael Brown	55
005	Mary Wilson	68
006	Jennifer Garcia	75
007	Elizabeth Martinez	92
008	Patricia Anderson	58
...
015	David Taylor	45

The goal is to remove records where the grade is below 60 and to view all students' records in the *student_grades* table.

Click on **Save & Run** to see the code run.



```
-- Remove records where  
the value in a column
```

(remove_records.html)

This example uses the following programming plans:

- Plan 1: Remove Records (remove_records.html)
- Plan 2: View Records (view_records.html)



Remove Records: Remove records where the value in a column meets a condition

```
DELETE FROM {table}
WHERE {condition};
```

► Show What To Write In Changeable Areas

Show Examples

Show Template

View Records: Select and view columns from the specified table


```
SELECT {column_or_columns}
FROM {table};
```

► Show What To Write In Changeable Areas

Show Examples

Show Template

Q-2: Click on “Show Examples” three times each for the plans above. Did you notice anything?

 First, complete the first two questions (Q1a and Q1b) on your worksheet. Then, click on the arrow on the bottom right to continue.



(start_here.html)

You have attempted of activities on this page



(remove_records.html)

```
{{ if request.application == 'runestone: }} {{ pass }}
```



(start_here.html)



(remove_records.html)

```
{{ if response.serve_ad and settings.adsenseid: }} {{ pass }}  
{{ if settings.num_banners > 0 and settings.show_rs_banner: }}
```

Please Support Runestone

(/runestone/default/donate?banner={{=banner_num}})

```
{{ pass }}
```



Plan 1: Remove Records

Remove Records: Remove records where the value in a column meets a condition

```
DELETE FROM {table}  
WHERE {condition};
```

► Show What To Write In Changeable Areas

Show Examples

Show Template

This plan is used to remove some records permanently from the table, using a condition to decide which records will be deleted.

Plan 1 - When to use this plan?

This plan is used when you want to remove records from the table using some criteria, such as dropping all student grades below some threshold, removing records that is missing a value in the specified field, or simply processing records that are marked for deletion.

Plan 1 - What parts can be customized to use this plan?

To use this plan, replace the table with the name of your table, and specify a condition using fields (columns) such that each record (row) that satisfies that condition will be deleted.

Plan 1 - Exercises



(integrated_3.html)



(view_records.html)

Q-1: Rearrange the SQL code blocks to remove records from the 'student_grades' <

```
DELETE FROM student_grades
---
WHERE grade < 60
---
WHERE COLUMN IS NULL #distractor
---
WHERE column=1 #distractor
---
WHERE COLUMN < 2 #distractor
```



Q-2: You are working with a database containing student grades. Students who have taken the first midterm have a value under the field 'mt1'. This field is empty for students who have not taken the exam (denoted by the special value NULL). You need to remove records for students who have not taken the exam. Which SQL statement correctly achieves this task?

- DELETE FROM student_grades WHERE mt1 IS NULL;
- Correct!
- DELETE FROM student_grades WHERE grade IS NULL;
- This option checks the 'grades' column for non-existent (NULL) values, but not the mt1 column.
- DELETE FROM student_grades WHERE mt1 = 0;
- This option removes records with a grade exactly equal to 0, but not for students who do not have any value.

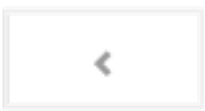
If you completed all the activities on this page, click on the arrow on the bottom right to continue.

You have attempted of activities on this page

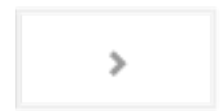
© Copyright 2013-2020 Runestone Interactive LLC. Created using Runestone (<http://runestoneinteractive.org/>) 5.3.0.

| [Back to](#)

{{ if request.application == 'runestone:{}'. }} {{ pass }}



(integrated_3.html)



(view_records.html)


```
{{ if response.serve_ad and settings.adsenseid: }} {{ pass }}  
{{ if settings.num_banners > 0 and settings.show_rs_banner: }}
```

Please Support Runestone

(/runestone/default/donate?banner={{=banner_num}})

```
{{ pass }}
```



Plan 2: View Records

View Records: Select and view columns from the specified table

```
SELECT [column_or_columns]  
FROM [table];
```

► Show What To Write In Changeable Areas

Show Examples

Show Template

The most fundamental query we will learn about is viewing records from the table using SELECT. This query will return a list of records (rows). For each record, you can see values for the fields (columns) that you specify in the query.

Plan 2 - When to use this plan?

Use this plan when you want to **view** some values in the database without modifying the table.

Plan 2 - What parts can be customized to use this plan?

To use this plan, replace the table with the table you want to get records (rows) from, and specify the fields (columns) you want to look for.

Plan 2 - Exercises

Complete the query to show the values of student_id for all records in the table student_grades.

```
SELECT  Enter the appropr  
FROM student_grades;
```

Check Me

Q-1: In the context of a database system where you want to view student records, the following SQL statement correctly selects all columns from the student_grades table: `SELECT * FROM student_grades;`

- True
- Correct!
- False
- The SQL statement is correct. Using '*' in the SELECT clause retrieves all columns from the specified table, which in this case is 'student_grades'.

If you completed all the activities on this page, click on the arrow on the bottom right to continue.

You have attempted of activities on this page

© Copyright 2013-2020 Runestone Interactive LLC. Created using Runestone (<http://runestoneinteractive.org/>) 5.3.0.

| [Back to](#)

{{ if request.application == 'runestone': }} {{ pass }}



(remove_records.html)



(integrated_2.html)

```
{{ if response.serve_ad and settings.adsenseid: }} {{ pass }}  
{{ if settings.num_banners > 0 and settings.show_rs_banner: }}
```

Please Support Runestone

(/runestone/default/donate?banner={{=banner_num}})

```
{{ pass }}
```



Example 2: Awarding Grammy's and finding recent Grammy nominee songs

In this example, we are managing a database of songs and their artists. Recently, several artists have won Grammy awards, and we need to update our records to reflect their new status as Grammy winners. The database contains columns such as 'artist_name', 'song_title', 'release_date', and 'award_status'. We will update the 'award_status' to 'Awarded' for all songs by artists who have won a Grammy.

Additionally, to showcase the most recent Grammy-winning songs, we will order the records by their release date in descending order. This allows us to present the latest award-winning songs at the top.

Below is an example of the table structure:

artist_name	song_title	release_date	award_status
Taylor Swift	Anti-Hero	2023-05-01	Nominated
Harry Styles	As It Was	2023-02-15	Awarded
Adele	Easy On Me	2022-11-12	Nominated
Lizzo	About Damn Time	2022-09-20	Awarded
Beyoncé	Break My Soul	2022-07-29	Awarded

By employing the plans 'Update Records Conditionally' and 'Order Records', we effectively maintain and display our data in a way that highlights recent achievements in the music industry.

Click on **Save & Run** to see the code run.

-- Update the value of a
column in all records



This example uses the following programming plans:

(view_records.html)

(update_records_conditionally.html)

- Plan 3: Update Records Conditionally (update_records_conditionally.html)
- Plan 4: Order Records (order_records.html)

Update Records Conditionally: Update the value of a column in all records meeting a condition

```
UPDATE {table}
SET {column_to_update} = {new_value}
WHERE {column_to_check} = {condition_value};
```

► Show What To Write In Changeable Areas

Show Examples

Show Template

Order Records: View records sorted in a given order

```
SELECT * FROM {table}
ORDER BY {column_to_sort_by} {order_direction};
```

► Show What To Write In Changeable Areas

Show Examples

Show Template

Q-2: Click on “Show Examples” three times each for the plans above. Have you noticed any other values that could be used in these plans?

Click on the arrow on the bottom right to continue.



You have attempted of activities on this page



```
{{ if response.serve_ad and settings.adsenseid: }} {{ pass }}  
{{ if settings.num_banners > 0 and settings.show_rs_banner: }}
```

Please Support Runestone

(/runestone/default/donate?banner={{=banner_num}})

```
{{ pass }}
```



Plan 3: Update Records Conditionally

Update Records Conditionally: Update the value of a column in all records meeting a condition

```
UPDATE {table}  
SET {column_to_update} = {new_value}  
WHERE {column_to_check} = {condition_value};
```

► Show What To Write In Changeable Areas

Show Examples

Show Template

This is a plan is used to update the values of some fields (columns) in existing records (rows). Only records that meet a certain condition is updated.

Plan 3 - When to use this plan?

This plan is used when you want to update the values of some fields in the table, but only for records that meet a certain condition. For example, you may want to update the status of all users who have not verified their email address, or update the location of all users who have not logged in for a certain period.

Plan 3 - What parts can be customized to use this plan?

To use this plan, replace the table name with the name of your table, and specify the fields (columns) that you want to update, the new values that you want to set, and the condition that must be met for the records to be updated.

Plan 3 - Exercises

Q-1: You are working in a database with user emails. You need to mark accounts that have used a test

email address for deletion. One of your colleagues shared the following query, saying that you can use this as a template. Click on the field(s) that you would need to change to check if the email column in the given table is 'test@email.com'.

```
UPDATE users
SET marked_for_deletion = true
WHERE name = Bob;
```

Q-2: In a database of customer records, you want to update all records with an email of 'test@email.com (mailto:'test%40email.com)'' by setting their 'column_name' to 'a_new_value'. The query `UPDATE table_name SET column_name = 'a_new_value' WHERE column1 = 'test@email.com (mailto:'test%40email.com)';` will achieve this goal.

- True
- Correct!
- False
- Ensure that the column being checked is correctly specified and that the new value is being set to the desired column.

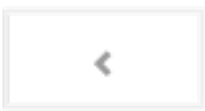
If you completed all the activities on this page, click on the arrow on the bottom right to continue.

You have attempted of activities on this page

© Copyright 2013-2020 Runestone Interactive LLC. Created using Runestone (<http://runestoneinteractive.org/>) 5.3.0.

| [Back to](#)

{{ if request.application == 'runestone': }} {{ pass }}



(integrated_2.html)



(order_records.html)

```
{{ if response.serve_ad and settings.adsenseid: }} {{ pass }}  
{{ if settings.num_banners > 0 and settings.show_rs_banner: }}
```

Please Support Runestone

(/runestone/default/donate?banner={{=banner_num}})

```
{{ pass }}
```



Plan 4: Order Records

Order Records: View records sorted in a given order

```
SELECT * FROM {table}  
ORDER BY {column_to_sort_by} {order_direction};
```

► Show What To Write In Changeable Areas

Show Examples

Show Template

This plan is used to see records (rows) sorted by some field (column), without actually modifying the table. The sorting will only apply to the output you see. It will not be ordered if you try to see it with a regular SELECT query.

Plan 4 - When to use this plan?

This plan may be used to see who are the top performers in a class, or which products in a database have the cheapest prices.

Plan 4 - What parts can be customized to use this plan?

To use this plan, complete the fields of the regular SELECT query as you saw in 'View Records', and then provide which field (column) to sort the records by, along with whether you want it ascending or descending.

Plan 4 - Exercises

Q-1: In a music database, to view all songs sorted by their release date in descending order, the following query is used: `SELECT * FROM songs ORDER BY release_date DESC ;`

- True
- Correct!
- False
- Remember that DESC is used to sort results in descending order, which means from newest to oldest.



Q-2: You are working with a music database and need to retrieve a list of songs sorted by their release date in descending order. Which SQL query would you use?

- `SELECT * FROM songs ORDER BY release_date DESC;`
- Correct!
- `SELECT * FROM songs ORDER BY release_date ASC;`
- This option sorts the songs in ascending order, showing the oldest songs first. You need to sort them in descending order.
- `SELECT * FROM songs ORDER BY column_name DESC;`
- This option uses 'column_name' instead of 'release_date'. Make sure to use the correct column to sort by.
- `SELECT * FROM table_name ORDER BY release_date DESC;`
- This option uses 'table_name' instead of 'songs'. Ensure you are querying from the correct table.

If you completed all the activities on this page, click on the arrow on the bottom right to continue.

You have attempted of activities on this page

© Copyright 2013-2020 Runestone Interactive LLC. Created using Runestone (<http://runestoneinteractive.org/>) 5.3.0.

| [Back to](#)

```
{{ if request.application == 'runestone: }} {{ pass }}
```



([update_records_conditionally.html](#))



([integrated_4.html](#))


```
{{ if response.serve_ad and settings.adsenseid: }} {{ pass }}  
{{ if settings.num_banners > 0 and settings.show_rs_banner: }}
```

Please Support Runestone

(/runestone/default/donate?banner={{=banner_num}})

```
{{ pass }}
```



Example 3: Tracking new movie releases

In this context, we are managing a database of movie releases. Each record in the database represents a movie, with details such as the movie title, release date, and genre. Our database needs to be updated regularly with new movie releases.

To track the status of movie industry, we also want to generate reports on how many movies have been released in each genre.

To achieve this, we will use two key SQL operations: inserting new records into the database and summarizing records to understand trends.

Below is a sample structure of the 'movies' table in the database:

Title	Release Date	Genre
Oppenheimer	2023-07-21	Drama
Barbie	2023-07-21	Comedy
Spider-Man	2021-12-17	Action
Everything	2022-03-25	Sci-Fi
Avatar	2009-12-18	Sci-Fi
Get Out	2017-02-24	Horror
La La Land	2016-12-09	Musical
Parasite	2019-10-11	Drama

We will demonstrate how to insert a new movie record into this table and how to summarize the number of movies for each genre. This will help us keep the database updated and generate insightful reports on movie genre trends.



(Click on **Save & Run** to see the code run.)



(add_new_record.html)

-- Insert a new record
(row) into the specified



This example uses the following programming plans:

- Plan 5: Add New Record (add_new_record.html)
- Plan 6: Summarize Records (summarize_records.html)

Add New Record: Insert a new record (row) into the specified table with values for specified columns

```
INSERT INTO {table} {field_names}  
VALUES {field_values}
```

► Show What To Write In Changeable Areas

Show Examples

Show Template

Summarize Records: Calculate aggregated summary statistics for the whole table, or for subsets of the table

```
SELECT {optional_column_to_group}, {aggregation_func}({column_to_process})  
FROM {table}  
GROUP BY {optional_column_to_group};
```

► Show What To Write In Changeable Areas

Show Examples

Show Template



(Q-2: Click on “Show Examples” three times each for the plans above. What are some other values for the changeable parts of these plans?)

Click on the arrow on the bottom right to continue.



You have attempted of activities on this page



© Copyright 2013-2020 Runestone Interactive LLC. Created using Runestone (<http://runestoneinteractive.org/>) 5.3.0.

| [Back to](#)

```
{{ if request.application == 'runestone': }} {{ pass }}
```



([order_records.html](#))



([add_new_record.html](#))

```
{{ if response.serve_ad and settings.adsenseid: }} {{ pass }}  
{{ if settings.num_banners > 0 and settings.show_rs_banner: }}
```

Please Support Runestone

(/runestone/default/donate?banner={{=banner_num}})

```
{{ pass }}
```



Plan 5: Add New Record

Add New Record: Insert a new record (row) into the specified table with values for specified columns

```
INSERT INTO {table} {field_names}  
VALUES {field_values}
```

► Show What To Write In Changeable Areas

Show Examples

Show Template

This is a plan to add new records (rows) into a table.

Plan 5 - When to use this plan?

This plan is used when you want to add more data to a table in your database.

Plan 5 - What parts can be customized to use this plan?

To use this plan, replace the name of the table, specify which fields will exist in the row you will insert, and then provide the values for those fields.

Plan 5 - Exercises



(integrated_4.html)



(summarize_records.html)

Q-1: Arrange the code blocks to insert a new record into the 'users' table with <

```
INSERT INTO users (name, email)
---
VALUES ('Alice', 'alice@illinois.edu')
---
INSERT INTO movies (title, release_date, genre) #distractor
---
VALUES ('Inception', '2010-07-16', 'Sci-Fi') #distractor
```

Q-2: You are tasked with adding a new record into the 'movies' table in a database using SQL. Which of the following SQL statements correctly adds a movie titled 'Inception', released on '2010-07-16', with the genre 'Sci-Fi'?

- INSERT INTO movies (title, release_date, genre) VALUES ('Inception', '2010-07-16', 'Sci-Fi');
- Correct!
- INSERT INTO users (name, email) VALUES ('Inception', '2010-07-16', 'Sci-Fi');
- Incorrect. This statement attempts to insert into the 'users' table, which is not where movie data belongs.
- INSERT INTO movies (name, email) VALUES ('Inception', '2010-07-16', 'Sci-Fi');
- Incorrect. This statement uses the wrong field names for the 'movies' table. It should be (title, release_date, genre).
- INSERT INTO users (title, release_date, genre) VALUES ('Inception', '2010-07-16', 'Sci-Fi');
- Incorrect. This statement attempts to insert into the 'users' table, which is not where movie data belongs.

If you completed all the activities on this page, click on the arrow on the bottom right to continue.

You have attempted of activities on this page

© Copyright 2013-2020 Runestone Interactive LLC. Created using Runestone (http://runestoneinteractive.org/) 5.3.0.

| Back to

{{ if request.application == 'runestone:{}'. }} {{ pass }}



(integrated_4.html)



(summarize_records.html)

```
{{ if response.serve_ad and settings.adsenseid: }} {{ pass }}  
{{ if settings.num_banners > 0 and settings.show_rs_banner: }}
```

Please Support Runestone

(/runestone/default/donate?banner={{=banner_num}})

```
{{ pass }}
```



Plan 6: Summarize Records

Summarize Records: Calculate aggregated summary statistics for the whole table, or for subsets of the table

```
SELECT {optional_column_to_group}, {aggregation_func}({column_to_process})  
FROM {table}  
GROUP BY {optional_column_to_group};
```

► Show What To Write In Changeable Areas

Show Examples

Show Template

This is a plan to view records summarized by some attribute. By summarize, we mean using some kind of aggregation function: such as taking the mean, maximum, minimum, or sum of the values. You can calculate this summary in subsets of the data (by using the GROUP BY part of the query), or you can calculate the summary values across all records.

Plan 6 - When to use this plan?

This plan is used when you want to get insights about your data by using statistical aggregation functions. You can use the 'GROUP BY' part to get results **for each subgroup**.

Plan 6 - What parts can be customized to use this plan?

To use this plan, replace the name of the table with your table, decide which function you want to use and which column(s) you want to summarize or see. If you want to calculate summary values over subsets, keep the GROUP BY line in the query and replace the column with a field. A separate group will be shown for each value of that field. For example, if you select AVG(quiz_grade) and if you group by letter_grade, you can see what was the average quiz grade for students who got an A, for students who got a B, etc.

Plan 6 - Exercises

Q-1: Arrange the following SQL query blocks to calculate the maximum value of


```
SELECT genre, MAX(column_2)
---
FROM movies
---
GROUP BY genre
---
SELECT title, AVG(column1) #distractor
---
GROUP BY column_name #distractor
```

Q-2: In the context of a movie database, using the plan 'Summarize Records', the SQL query `SELECT genre, AVG(rating) FROM movies GROUP BY genre;` correctly calculates the average rating for each genre.

- True
- Correct!
- False
- This query correctly uses the 'AVG' aggregation function on the 'rating' column and groups the results by 'genre', which is a valid approach to obtain the average rating for each genre.

Q-3: Using the 'Summarize Records' plan, the following query will correctly calculate the average rating of movies grouped by genre: `SELECT genre, AVG(rating) FROM movies GROUP BY genre;`

- True
- Correct!
- False
- Make sure to check if the aggregation function and the column names used in the query match the plan's template structure.

 First, complete the next two questions (Q2a and Q2b) on your worksheet. Then, if you completed all the activities on this page, click on the arrow on the bottom right to continue.

You have attempted of activities on this page

```
{{ if response.serve_ad and settings.adsenseid: }} {{ pass }}  
{{ if settings.num_banners > 0 and settings.show_rs_banner: }}
```

Please Support Runestone

(/runestone/default/donate?banner={{=banner_num}})

```
{{ pass }}
```



Exercise 1: Writing Queries for Ad Management

Let's assume that you are running the advertisement campaign for a drink company. You have multiple ads with different celebrities, and you keep track of who you worked with, the cost of each ad, and projected reach (how many users have seen the ad). Here's an example table:

image[Ad campaign with celebrity endorsements]

Celebrity	Ad Cost (USD)	Projected Reach
Taylor Swift	500,000	5,000,000
George Clooney	300,000	2,000,000
Emma Watson	150,000	1,000,000
Chris Hemsworth	200,000	1,500,000
Jennifer Lopez	400,000	3,000,000
Robert Downey Jr.	450,000	4,000,000

Q-1: You just started a new ad campaign with the rapper Kendrick Lamar. However

Order Records

Remove Records

Add New Record

Summarize Records #distractor

View Records #distractor

Update Records Conditionally #distractor



(summarize_records.html)



(codereading-1.html)

Q-2: You have already started your campaign with Kendrick Lamar. Your manager

```
SELECT AVG('Ad Cost')
---
FROM 'Ads'
---
GROUP BY 'Celebrity' #distractor
---
DELETE FROM 'Ad Cost' #distractor
---
WHERE 'Ad Cost' = AVG('Ad Cost') #distractor
---
ORDER BY 'Ad Cost' #distractor
```



First, complete the next question (Q3) on your worksheet. Then, click on the arrow on the bottom right to continue.

You have attempted of activities on this page

© Copyright 2013-2020 Runestone Interactive LLC. Created using Runestone (<http://runestoneinteractive.org/>) 5.3.0.

| [Back to](#)

{{ if request.application == 'runestone': }} {{ pass }}



([summarize_records.html](#))



([codereading-1.html](#))

```
{{ if response.serve_ad and settings.adsenseid: }} {{ pass }}  
{{ if settings.num_banners > 0 and settings.show_rs_banner: }}
```

Please Support Runestone

(/runestone/default/donate?banner={{=banner_num}})

```
{{ pass }}
```



Exercise 2: Managing Spotify Databases

You are working for Spotify. One of your coworkers shared the following queries with you, and asked you to interpret the output in a report. However, they did not comment their code, and it looks like they are offline, so you need to figure out what this does.

Look at the code and describe what the queries might be doing.

```
INSERT INTO Users  
(username, email,
```

Q-2: Enter your description here.

If you completed all the activities on this page, click on the arrow on the bottom right to continue.

You have attempted of activities on this page

© Copyright 2013-2020 Runestone Interactive LLC. Created using Runestone (<http://runestoneinteractive.org/>) 5.3.0.

| [Back to](#)

```
{{ if request.application == 'runestone': }} {{ pass }}
```



(codewriting-1.html)



(end_here_comparison.html)

{{ if response.serve_ad and settings.adsenseid: }} {{ pass }}
{{ if settings.num_banners > 0 and settings.show_rs_banner: }}

Please Support Runestone

(/runestone/default/donate?banner={{=banner_num}})

{{ pass }}



Before You Complete The Activity - 1

Please rate how much you agree with the each statement. Your responses will be anonymized.

I am interested in learning how to manage data and run SQL queries on databases.

- 1. Strongly Disagree
- 2. Disagree
- 3. Somewhat Disagree
- 4. Neither Agree nor Disagree
- 5. Somewhat Agree
- 6. Agree
- 7. Strongly Agree

I am confident in my ability to write or edit SQL queries to analyze datasets.

- 1. Strongly Disagree
- 2. Disagree
- 3. Somewhat Disagree
- 4. Neither Agree nor Disagree
- 5. Somewhat Agree
- 6. Agree
- 7. Strongly Agree

If you completed all the activities on this page, click on the arrow on the bottom right to continue.

You have attempted of activities on this page

© Copyright 2013-2020 Runestone Interactive LLC. Created using Runestone (http://runestoneinteractive.org/) 5.3.0.

| Back to

{{ if request.application == 'runestone': }} {{ pass }}



(codereading-1.html)



(end_here_usability.html)

{{ if response.serve_ad and settings.adsenseid: }} {{ pass }}
{{ if settings.num_banners > 0 and settings.show_rs_banner: }}

Please Support Runestone

(/runestone/default/donate?banner={{=banner_num}})

{{ pass }}



Before You Complete The Activity - 2

Rate how much you agree with the given statement based on your experience with this interface.

Overall, I am satisfied with how easy it is to use this system.

- 1. Strongly Disagree
- 2. Disagree
- 3. Somewhat Disagree
- 4. Neither Agree nor Disagree
- 5. Somewhat Agree
- 6. Agree
- 7. Strongly Agree

I felt comfortable using this system.

- 1. Strongly Disagree
- 2. Disagree
- 3. Somewhat Disagree
- 4. Neither Agree nor Disagree
- 5. Somewhat Agree
- 6. Agree
- 7. Strongly Agree

I believe I learned this material effectively using this system.

- 1. Strongly Disagree
- 2. Disagree
- 3. Somewhat Disagree
- 4. Neither Agree nor Disagree
- 5. Somewhat Agree
- 6. Agree
- 7. Strongly Agree

The information provided with the system was effective in helping me complete the tasks & scenarios.

- 1. Strongly Disagree
- 2. Disagree
- 3. Somewhat Disagree
- 4. Neither Agree nor Disagree
- 5. Somewhat Agree

- 6. Agree
- 7. Strongly Agree



I liked using the interface of this system.

- 1. Strongly Disagree
- 2. Disagree
- 3. Somewhat Disagree
- 4. Neither Agree nor Disagree
- 5. Somewhat Agree
- 6. Agree
- 7. Strongly Agree

Q-6: If you would to participate in an OPTIONAL 30-minute interview about your experience, please provide your email address below.



Complete the last two questions (Q4 and Q5) on your worksheet. You have reached the end of the online activities. If you completed the online activities and your reflection sheet, you can check in with your TA. If not, click here to go back to the table of contents. ([/ns/books/published/cs102sql/index.html](#))

You have attempted of activities on this page

© Copyright 2013-2020 Runestone Interactive LLC. Created using Runestone (<http://runestoneinteractive.org/>) 5.3.0.

| [Back to](#)

```
{{ if request.application == 'runestone': }} {{ pass }}
```



([end_here_comparison.html](#))