Unit 9: Databases & The Modern Web

Assessment #8

Assigned: 03/08/23

Due: 03/08/23

Completed: 03/08/23

**LMS:** <https://lms.grandcircus.co/mod/assign/view.php?id=22874>

**Google Doc:** [https://docs.google.com/document/d/1oMsDXszDrouHZ17fo2CF9mciisKKgbzU\_ukNP0nwuVY/preview#](https://docs.google.com/document/d/1oMsDXszDrouHZ17fo2CF9mciisKKgbzU_ukNP0nwuVY/preview)

**GitHub:** <https://github.com/gc-submissions/assessment-8-clintmsmith>

This assessment has two parts: Part 1 - SQL is worth 4 points. Part 2 - MongoDB is worth 6 points.

**PART 1 - SQL**

**Overview:** Within PGAdmin, run the provided setup code to create the initial tables. Then complete SQL code for the four numbered bullets below.

**Setup:**

First, copy-paste the code from part1-setup.sql into pgAdmin and run it. It creates two tables: **students** and **assignments**. It also adds five rows to the **students** table, but the **assignments** table starts out empty. This provides the starting point for the questions below.

**Submission:**

In the provided SQL file part1-answers.sql, paste the SQL code for each of the four numbered bullets below. You do not need to submit the results of the queries, just the SQL code itself.

**SQL Questions:**

1. Select all students that have both **firstname** starting with **"J"** and **id** greater than **3**. (In this case, only Javier would match.)

SELECT \* FROM students

WHERE firstname LIKE ‘J%’

AND id > 3

1. Update the student with the **id** of **4** to have a **lastname** of **Chirpus**.

UPDATE students

SET lastname = ‘Chirpus’

WHERE id = 4

1. Add these three assignments. You may use multiple SQL statements. (Note: **student\_id** is a foreign key reference to **students**. This is a many-to-one relationship: many assignments to one student.)
   * student: **Jane**, title: **Geography Quiz**, score: **85**
   * student: **Jane**, title: **US States Worksheet**, score: **100**
   * student: **Javier**, title: **Geography Quiz**, score: **92**

INSERT INTO assignments (title, score, student\_id)

VALUES (‘Geography Quiz’, 85, 1), (‘US States Worksheet’, 100, 1), (‘Geography Quiz’, 92, 4)

1. In one SQL statement, get the first and last names of all students that scored **above 90** on the **Geography Quiz**. (Include ***only*** the **firstname** and **lastname** columns in the result.)

SELECT firstname, lastname

FROM students

INNER JOIN assignments

ON students.id = assignments.student\_id

WHERE assignments.title = ‘Geography Quiz’ AND assignments.score > 90

**PART 2 - MONGODB**

**Submission:**

In the provided JavaScript file part2-answers.js, paste the MongoDB command for each of the six numbered bullets below. You do not need to submit the results of the queries, just the command itself.

**MongoDB Questions:**

1. Using one MongoDB command, add the following data to a collection named **ticketsales**. Let MongoDB generate IDs for you. (You won't lose points for typos in the data.)

|  |  |  |
| --- | --- | --- |
| **show** | **section** | **sold** |
| Aretha Franklin | Main Floor | 200 |
| Aretha Franklin | Balcony | 100 |
| The Temptations | Main Floor | 250 |
| The Temptations | Balcony | 80 |
| The Supremes | Main Floor | 250 |
| The Supremes | Balcony | 120 |
| The Supremes | Box Seats | 0 |

db.ticketsales.insertMany([

{‘show’: ‘Aretha Franklin’, ‘section’: ‘Main Floor’, ‘sold’: 200},

{‘show’: ‘Aretha Franklin’, ‘section’: ‘Balcony’, ‘sold’: 100},

{‘show’: ‘The Temptations’, ‘section’: ‘Main Floor’, ‘sold’: 250},

{‘show’: ‘The Temptations’, ‘section’: ‘Balcony’’, ‘sold’: 80},

{‘show’: ‘The Supremes’, ‘section’: ‘Main Floor’, ‘sold’: 250},

{‘show’: ‘The Supremes’, ‘section’: ‘Balcony’’, ‘sold’: 120},

{‘show’: ‘The Supremes’, ‘section’: ‘Box Seats’, ‘sold’: 0},

])

1. Add **10** to tickets **sold** for the **Balcony** at **The Temptations** show. Your command must use "Balcony" and "The Temptations". Do not reference the ID.

db.ticketsales.updateOne( {show: ‘The Temptations’, section: ‘Balcony’}, { $inc: {sold: 10} } )

1. Using one MongoDB command, find all data for shows by **The Supremes** or **The Temptations**.

db.ticketsales.find({ $or: [ {show: ‘The Supremes’}, {show: ‘The Temptations’} ] })

1. Use projection to show ***only*** the **number sold** for **Aretha Franklin** on the **Main Floor**.

db.ticketsales..aggregate( { $match: { show: ‘Aretha Franklin’, section: ‘Main Floor’ } }, { $project: {number\_sold: ‘$sold’} } )

1. Delete the document for **Box Seats** at **The Supremes**. Reference the document **by its ID**.

db.ticketsales.deleteOne({ db.ticketsales.findOne({show: ‘The Supremes’, section: ‘Box Seats’}).\_id })

1. Write a query that will automatically find the document with the least amount of tickets sold.

db.ticketsales.aggregate([ { $sort: {sold: 1} }, { $limit: 1 } ])