**SQL**

**Relational databases** store data in tables, which are connected to each other in a variety of different ways. Tables contain columns and rows of information, with each column specifying the data type of the information within, and each row having a unique key. Relational databases provide flexibility for both expansion of the database and modification of the relationships between the tables as things change.

In order to access our information, we need to use a tool that can talk to a relational database. **Structured Query Language** or **SQL** is the main tool used by programmers to work with these data structures. SQL is a **Relational Database Management System** or **RDMS**. We can use SQL to perform many essential operations on a database, such as adding and removing data.

**MYSQL**

MySQL is open-source and user-friendly. It is a type of SQL.

**MYSQL DataTypes**

Numbers: INT, BIGINT, MEDIUMINT, double

String: varchar(max. No of characters) varchar(250)

Boolean: BOOL and BOOLEAN 0 (false) non zero values like 1 for true

**Knowledge Check**

Which of the following are supported data types in MySQL? Select ALL that are correct.

1. **TINYINT**
2. **INTEGER**
3. **DOUBLE**
4. **VARCHAR**
5. **BOOLEAN**
6. **STRING**

**Answer: ALL**

**SQL Queries**

* When we want to perform an operation on a database, we write a SQL query.
* Each query produces a **result set** containing the requested data.
* **CRUD** or *Create*, *Read*, *Update*, *Delete* represents the four major operations we perform when we work with data.

#### 

#### 

#### 

#### 

#### **Creating a Table from Scratch - creating empty table**

**CREATE** **TABLE** li\_wedding (

guest\_id **INT**,

last\_name **VARCHAR**(**255**),

first\_name **VARCHAR**(**255**),

attending **BOOL**,

diet **VARCHAR**(**255**)

);

This query creates a new table in our database called **li\_wedding**. This table contains five columns: **guest\_id** of type **INT**, **last\_name** of type **VARCHAR(255)**, **first\_name** of type **VARCHAR(255)**, **attending** of type **BOOL**, **diet** of type **VARCHAR(255)**. When we use **CREATE TABLE** to create a table from scratch, we will end up with an empty table.

#### **Creating a Table from Another Table**

We can also create a table from another table. When doing so, the new table will also include any existing data from the original table.

**CREATE** **TABLE** johnson\_vow\_renewal

**AS** **SELECT** guest\_id, last\_name, first\_name, attending, diet

**FROM** johnson\_wedding

**WHERE** attending = **1**;

**Adding a Row**

Whether we have created a table from scratch or are working with an existing table, we may need to add a row of data. To do so, we need to use an **INSERT INTO** statement.

**INSERT** **INTO** johnson\_vow\_renewal

**VALUES** (**185**, "Johnson", "Eliza", **1**, "Vegan");

This query adds a row for Eliza to the **johnson\_vow\_renewal** table in our database.

**INSERT** **INTO** johnson\_vow\_renewal (guest\_id, last\_name, first\_name)

**VALUES** (**186**, "Johnson", "Felicity");

When we use this method, any column that doesn’t have a specified value for the new row will have a **null** value.

**Adding a column**

To add a column, we need to start with an ALTER TABLE statement. ALTER TABLE can be used to perform different operations, so in our case, we will also need to specify that we want to ADD a column.

**ALTER** **TABLE** li\_wedding **ADD** can\_drink **boolean**;

This adds the **can\_drink** column to the **li\_wedding** table, but it does *not* fill that column with values. We will need to update each guest’s entry in the table once we confirm how old they are.

## 

**Read**

When reading data, we don’t want to modify anything, we just want to know what is there. In order to get information from a table, we need to use a **SELECT** statement.

**SELECT column\_name\_1, column\_name\_2, ... FROM table\_name**

**WHERE some conditional is true.**

Example,

**SELECT** last\_name, first\_name **FROM** li\_wedding **WHERE** (attending = **1**) **AND** (diet = "vegetarian");

To get all rows and columns in a table,

**SELECT** \* **FROM** li\_wedding;

**Update**

**UPDATE johnson\_vow\_renewal SET diet="vegetarian" WHERE guest\_id=185;**

Now if we use a **SELECT** statement, we can confirm that we have properly updated the record.

**SELECT** \* **FROM** johnson\_vow\_renewal **WHERE** guest\_id=**185**

**UPDATE** johnson\_vow\_renewal **SET** diet="vegetarian", first\_name="Elizabeth" **WHERE** guest\_id=**185**;

**Warning**

**If you do not include a condition with WHERE, all records in the table will be updated**

**Delete**

Deleting a record permanently removes it from the table! Proceed with caution with removing records!

**DELETE** **FROM** johnson\_vow\_renewal **WHERE** guest\_id=**107**;

To confirm,

**SELECT** \* **FROM** johnson\_vow\_renewal **WHERE** guest\_id=**107**;

| **Question**  What does the following query do?  **SELECT EventID FROM EventsMaster WHERE (Month=07);**   1. Returns the event id from a table called EventsMaster for all events in 7 months of the year. 2. Returns the event id for all events in a table called EventsMaster for the month of July. 3. Returns the event id for all events in a table called EventsMaster for the month of June.   **Answer:**  **B** Returns the event id for all events in a table called EventsMaster for the month of July. |
| --- |

| **Question**  Mary has hired another event planner, Leah. We now need to create a table for the events that Leah is going to be planning. We also need to add a row for her first clients, Tate and Carlos. Does the following query accomplish this task?   | **1**  **2**  **3**  **4**  **5**  **6**  **7**  **8**  **9**  **10** | **CREATE TABLE LeahEvents (**  **EventID int,**  **EventName varchar(255),**  **Month int,**  **Day int,**  **Year int**  **);**  **INSERT INTO LeahEvents VALUES (256, "SmithWedding", 08, 08,2021);** | | --- | --- |   **Answer:** Yes |
| --- | --- | --- |

**SQL Joins**

* A **join** combines two tables into one result set. We can use joins when we want to query two tables at the same time. Whenever we join two tables, we have to specify the condition upon which the tables need to be joined.

In SQL, there are four different types of joins:

1. Inner Join
2. Left Outer Join
3. Right Outer Join
4. Full Outer Join

**General syntax for Join**

**SELECT** column\_name\_1, column\_name\_2, .... **FROM** table\_a

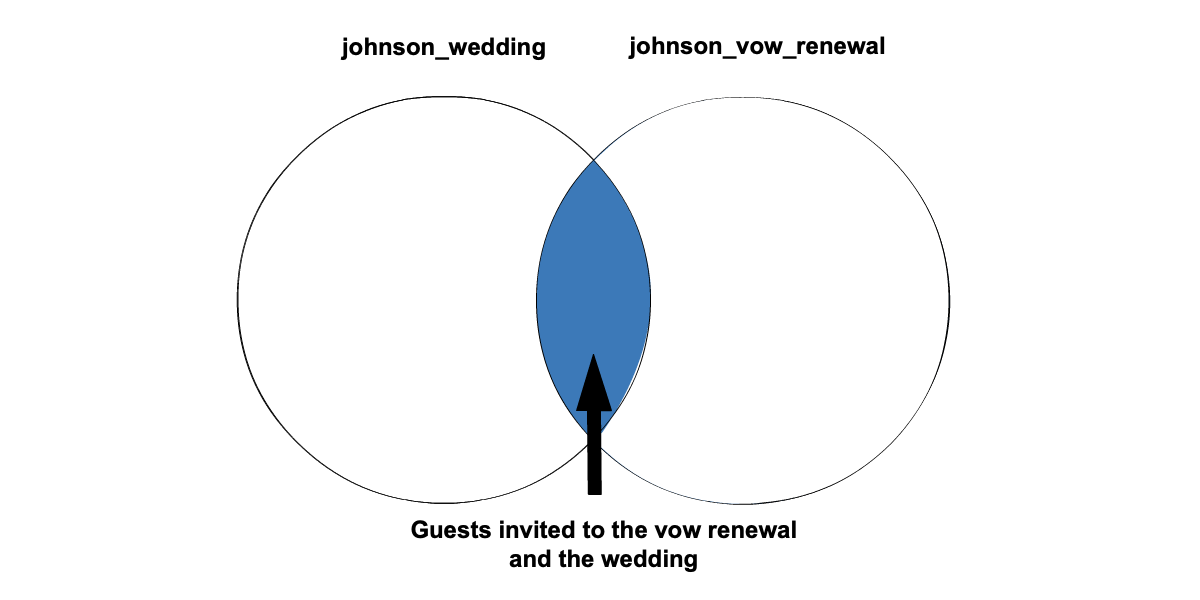
TYPEOFJOIN **JOIN** table\_b **ON** table\_a.column\_name\_1 = table\_b.column\_name\_1;

**Inner Join**

* Joining two tables with an **inner join** produces a result set that only includes the values that are present in *both* tables.
* Example

**SELECT** last\_name, first\_name **FROM** johnson\_vow\_renewal

**INNER** **JOIN** johnson\_wedding **ON** johnson\_vow\_renewal.guest\_id = johnson\_wedding.guest\_id;

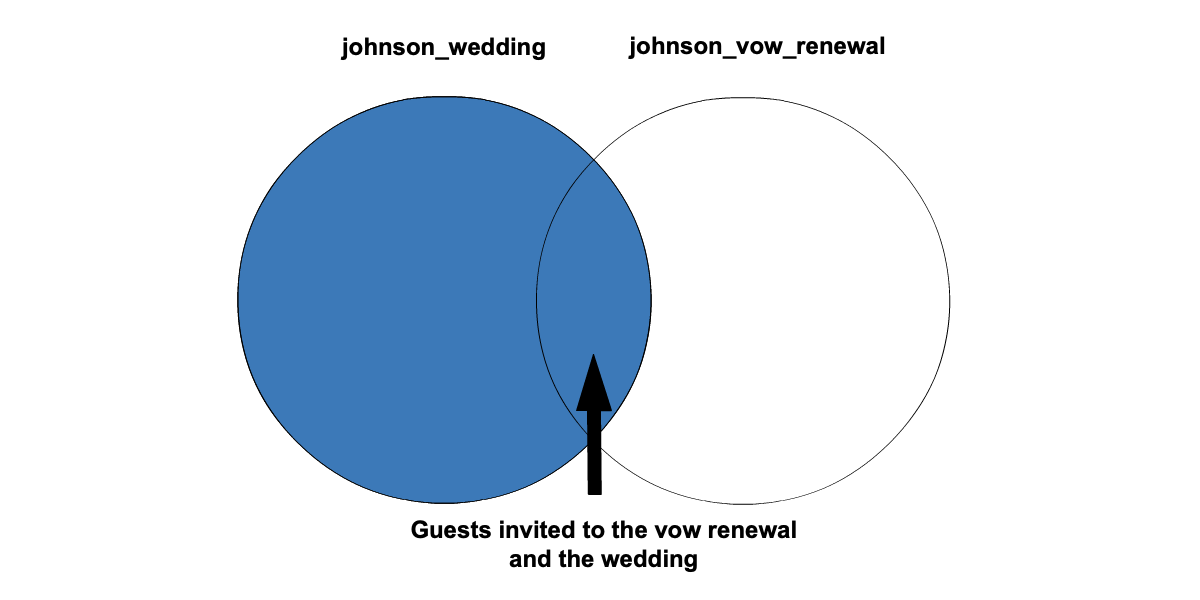


**Left Outer Join**

* Joining two tables with a **left outer join** gives us a result set which includes all values in the left table and any matching records from the right table.
* Example

**SELECT** last\_name, first\_name **FROM** johnson\_wedding

**LEFT** **JOIN** johnson\_vow\_renewal **ON** johnson\_wedding.guest\_id = johnson\_vow\_renewal.guest\_id;

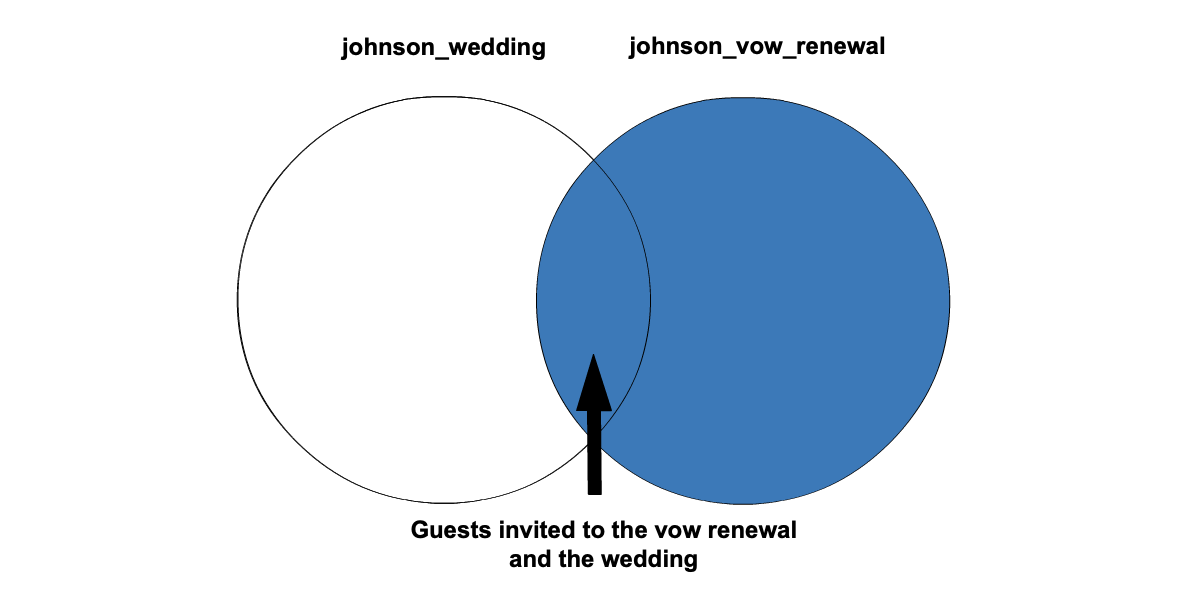


**Right Outer Join**

* Joining two tables with a **right outer join** gives us a result set that includes all values in the right table and any matching records from the left table.
* Example

**SELECT** last\_name, first\_name **FROM** johnson\_wedding

**RIGHT** **JOIN** johnson\_vow\_renewal **ON** johnson\_wedding.guest\_id = johnson\_vow\_renewal.guest\_id;



**Full Outer Join**

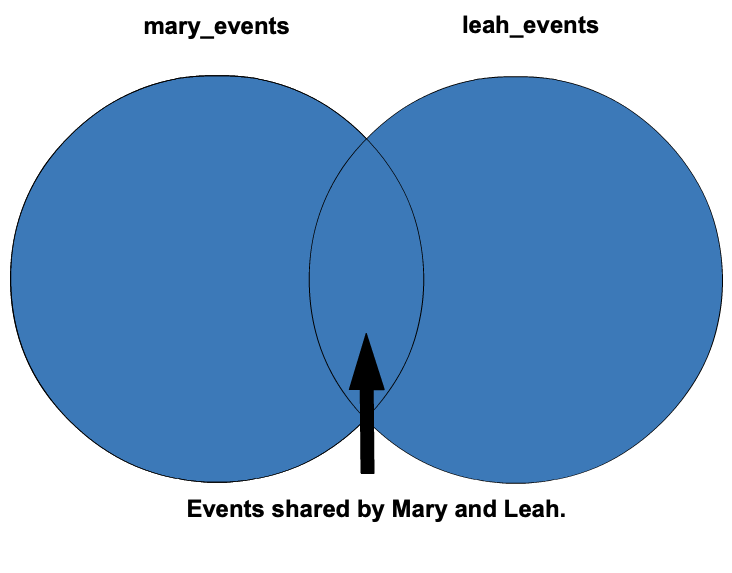
* Joining two tables with a **full outer join** gives us a result set that includes all records from both tables. Full outer joins are important to SQL, but the syntax is not supported in MySQL.
* Example

**SELECT** \* **FROM** mary\_events

**FULL** **OUTER** **JOIN** leah\_events

**ON** mary\_events.**month** = leah\_events.**month**

**WHERE** mary\_events.**month** = **08**;



If you do want to try out a full outer join, the syntax to simulate it looks some like this:

**SELECT** \* **FROM** table\_a **LEFT** **JOIN** table\_b **ON** table\_a.column\_name\_1 = table\_b.column\_name\_1

**UNION**

**SELECT** \* **FROM** table\_a **RIGHT** **JOIN** table\_b **ON** table\_a.column\_name\_1 = table\_b.column\_name\_2;

**UNION** is used to bring together the result sets of 2 **SELECT** queries. Check out the [documentation](https://dev.mysql.com/doc/refman/8.0/en/union.html) for more information on how **UNION** works

**Qustion 1**: an inner join gives all of the records in both tables - true

**Question 2:**

What bugs are in this SQL query? Select all that apply.

**SELECT FROM** johnson\_vow\_renewal **OUTER** **JOIN** johnson\_weding **ON** johnson\_vow\_renewal.guest\_id = johnson\_wedding.guest\_id

1. Nothing specified after **SELECT**.
2. **johnson\_vow\_renewal** is spelled wrong.
3. The type of join is not specified.
4. **johnson\_wedding** is spelled wrong.
5. **ON** is in the wrong place.
6. There isn’t a semicolon on the end of the SQL query.

Answer: a and c