Workshop: Intro to SQL

DSI SUDS Skills Day

Ciara Zogheib

Hello!

- Ciara Zogheib (She/Her)
- Starting my PhD in information this fall
- Working as a data scientist with government for ~2 years
- First learned SQL in a tiny island village in 2 weeks

01 Background Info

What’s an RDBMS, what’s SQL, SQL ‘dialects’

02 Best Practices

Things to keep in mind when we’re writing queries

03 Playing with SQL

Basic queries, exploring a sample database

Today's Goal:

Be able to go into a relational database and pull out specific subsets of data using SQL.

01

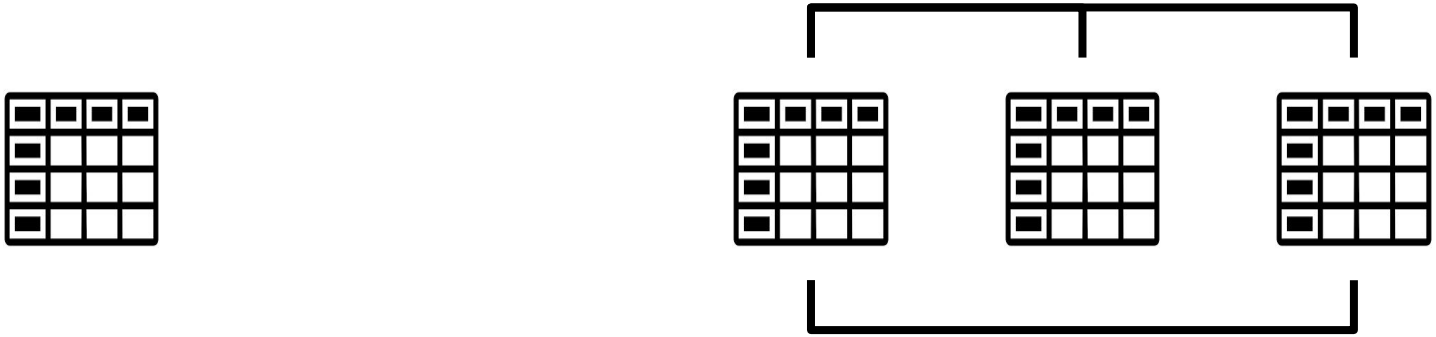
Background Info

Data

- We can think of data as symbols representing observed properties of objects, events, or the world around us
- Data can be numbers, words, stories, colours, sounds – any systematically collected and organized observation
- Unstructured data = documents or images, individual files
- Structured data = data in tables, Excel spreadsheet

Today, we focus on structured tabular data

Tables VS Databases



- A relational database management system (RDBMS) lets us store, view, and query multiple tables and the relationships between them
- There are many different types of RDBMS (e.g. MySQL, Oracle, SQLite, MS Access, etc.)

Related Tables

- Related tables are connected by primary and foreign keys
- Primary key = an attribute that uniquely identifies a row/record (e.g. student id number or an automatically generated incremental integer number)
- Foreign key = when a primary key is referenced in another table (e.g. the table of teachers' students would reference student id numbers)
- Primary and foreign keys help us normalize our databases

SQL

- SQL (Structured Query Language) lets us ‘query’ (access, extract from, and manipulate) the data in our RDBMS
- SQL can help us extract and combine data from multiple related tables at once, or with conditions applied
- Different RDBMSs sometimes use different SQL syntax, but a general understanding of SQL is easily transferable (different dialects, not different languages)

Today's Most Controversial Slide

How should you pronounce SQL?

...however you want.*

*within reason

SQL queries can be simple...

```
SELECT * FROM studentstable WHERE grade = 2;
```

...or complex

```
WITH tmp_1 AS
    (
    SELECT Calc1 =
        (  (SELECT TOP 1 DataValue
            FROM (
                    SELECT TOP 50 PERCENT DataValue
                    FROM SOMEDATA
                    WHERE DataValue IS NOT NULL
                    ORDER BY DataValue
                 ) AS A
            ORDER BY DataValue DESC
            ) +

            (SELECT TOP 1 DataValue
            FROM (
                    SELECT TOP 50 PERCENT DataValue
```

...and integrated with other code

```
def create_server_connection(host_name, user_name, user_password):
    connection = None
    try:
        connection = mysql.connector.connect(
            host=host_name,
            user=user_name,
            passwd=user_password
        )
        print("MySQL Database connection successful")
    except Error as err:
        print(f"Error: '{err}'")

    return connection
```

02

Best Practices

Best Practices

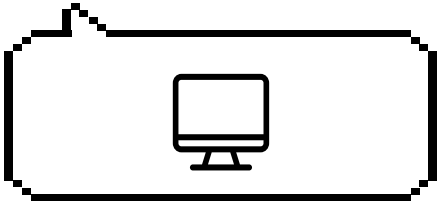
- Write SQL keywords uppercase and table/column names lowercase
  - SELECT column FROM table; **not** select column from table;
- Indent and use spaces to make long SQL queries readable
- Explicitly specify what columns you want to select
  - SELECT name, age FROM table; **not** SELECT * FROM table
- Consider using multiple queries instead of one monster query
  - When working with large quantities of data, multiple queries can slow processing and decrease performance – use judgment!
- Comment your code!

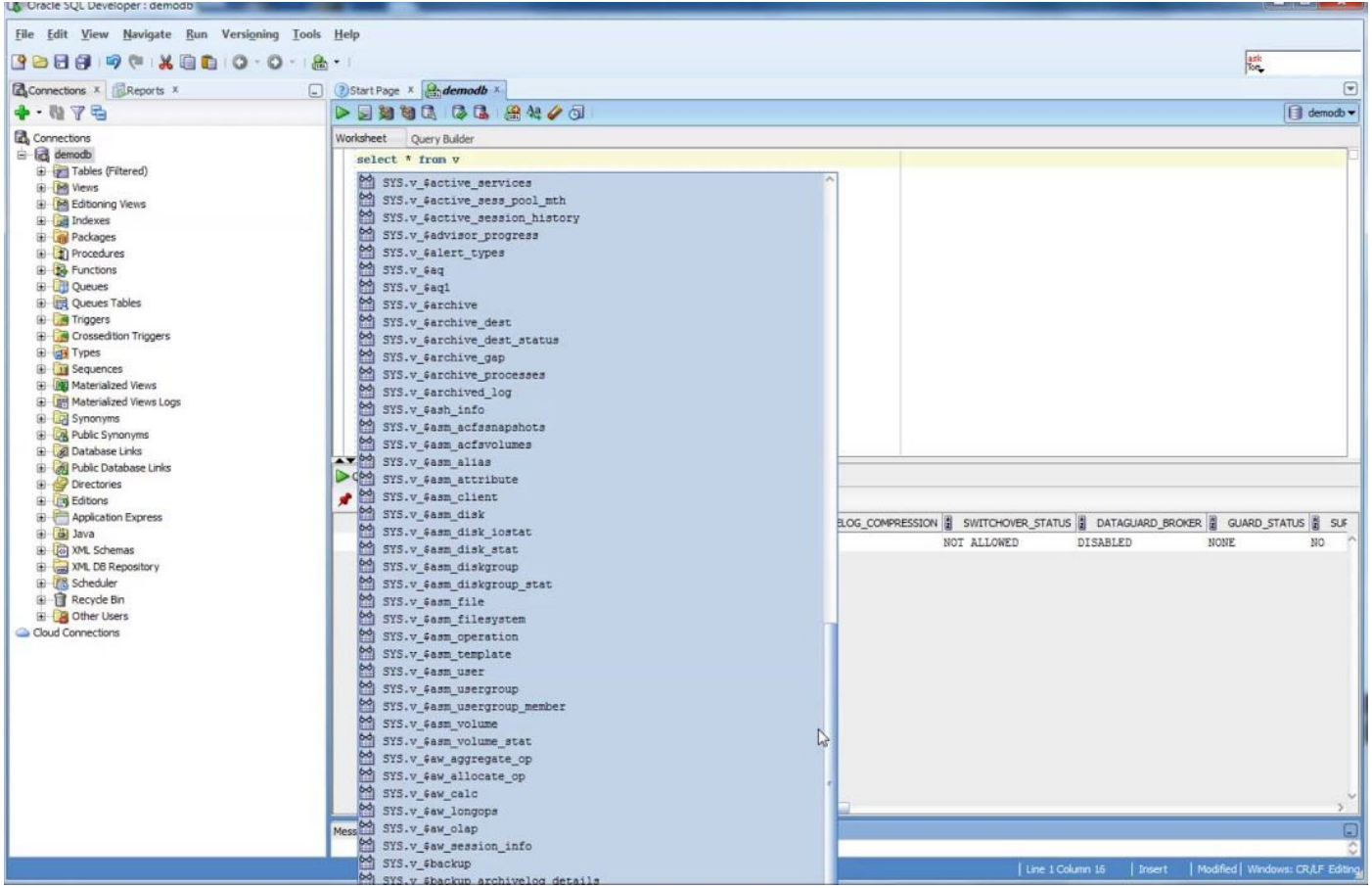
Best Practices

- Have a backup of your data if you are adding, removing, or changing data using SQL queries!
  - A copy of the table in the same database and/or
  - A copy of the table in a different backup database and/or
  - A backup of the entire database
- Double check your data after executing some change

03

Playing with SQL




Oracle SQL Developer : demodb
File Edit View Navigate Run Versioning Tools Help
Connections
Reports
Start Page
demodb
Worksheet
Query Builder
select * from v
SYS.v_$active_services
SYS.v_$active_sess_pool_mth
SYS.v_$active_session_history
SYS.v_$advisor_progress
SYS.v_$alert_types
SYS.v_$aq
SYS.v_$aq1
SYS.v_$archive
SYS.v_$archive_dest
SYS.v_$archive_dest_status
SYS.v_$archive_gap
SYS.v_$archive_processes
SYS.v_$archived_log
SYS.v_$ash_info
SYS.v_$asm_acfssnapshots
SYS.v_$asm_acfsvolumes
SYS.v_$asm_alias
SYS.v_$asm_attribute
SYS.v_$asm_client
SYS.v_$asm_disk
SYS.v_$asm_disk_iostat
SYS.v_$asm_disk_stat
SYS.v_$asm_diskgroup
SYS.v_$asm_diskgroup_stat
SYS.v_$asm_file
SYS.v_$asm_filesystem
SYS.v_$asm_operation
SYS.v_$asm_template
SYS.v_$asm_user
SYS.v_$asm_usergroup
SYS.v_$asm_usergroup_member
SYS.v_$asm_volume
SYS.v_$asm_volume_stat
SYS.v_$aw_aggregate_op
SYS.v_$aw_allocate_op
SYS.v_$aw_calc
SYS.v_$aw_longops
SYS.v_$aw_olap
SYS.v_$aw_session_info
SYS.v_$backup
SYS.v_$backup_archivelog_details
Connections
demodb
Tables (Filtered)
Views
Editioning Views
Indexes
Packages
Procedures
Functions
Queues
Queues Tables
Triggers
Crossedition Triggers
Types
Sequences
Materialized Views
Materialized Views Logs
Synonyms
Public Synonyms
Database Links
Public Database Links
Directories
Editions
Application Express
Java
XML Schemas
XML DB Repository
Scheduler
Recycle Bin
Other Users
Cloud Connections
LOG_COMPRESSION
SWITCHOVER_STATUS
DATAGUARD_BROKER
GUARD_STATUS
NOT ALLOWED
DISABLED
NONE
NO
Line 1 Column 16
Insert
Modified
Windows: CR/LF
Editing

Schema_Music.mwb - MySQL Workbench

Local instance 3306 | MySQL Model | EER Diagram

Management | Schemas

SCHEMAS

Filter objects

- Music
  - Tables
    - Albums
      - Columns
      - Indexes
      - Foreign Keys
      - Triggers
    - Artists
      - Columns
      - Indexes
      - Foreign Keys
      - Triggers
    - Genres
      - Columns
      - Indexes
      - Foreign Keys
      - Triggers
  - Views
  - Stored Procedures
  - Functions
- sys

Object Info | Session

Schema: Music

MySQL_Schema_Music_Example

Limit to 1000 rows

```
-- MySQL Script generated by MySQL Workbench
-- Mon May 30 11:25:32 2016
-- Model: New Model    Version: 1.0
-- MySQL Workbench Forward Engineering

SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0;
SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='TRADITIONAL,ALLOW_INVALID_DATES';

-- -----------------------------------------------------
-- Schema Music
-- -----------------------------------------------------

DROP SCHEMA IF EXISTS `Music` ;

-- -----------------------------------------------------
-- Schema Music
-- -----------------------------------------------------

CREATE SCHEMA IF NOT EXISTS `Music` DEFAULT CHARACTER SET utf8 ;
USE `Music` ;
```

100% | 45:1

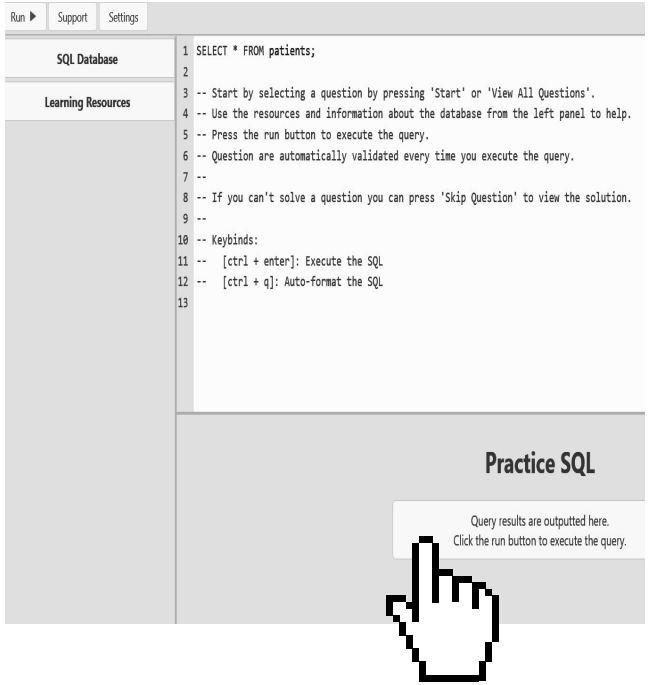
Action Output

	Time	Action	Response	Duration / Fetch Time
4	11:27:13	DROP SCHEMA IF EXISTS `Music`	0 row(s) affected, 1 warning(s): 1008 Can't drop data...	0.00034 sec
5	11:27:13	CREATE SCHEMA IF NOT EXISTS `...	1 row(s) affected	0.0012 sec
6	11:27:13	USE `Music`	0 row(s) affected	0.00030 sec
7	11:27:13	DROP TABLE IF EXISTS `Music`.`Artists`	0 row(s) affected, 1 warning(s): 1051 Unknown table...	0.00032 sec
8	11:27:13	CREATE TABLE IF NOT EXISTS `Mu...	0 row(s) affected	0.027 sec
9	11:27:13	DROP TABLE IF EXISTS `Music`.`Ge...	0 row(s) affected, 1 warning(s): 1051 Unknown table...	0.00037 sec
10	11:27:13	CREATE TABLE IF NOT EXISTS `Mu...	0 row(s) affected	0.026 sec
11	11:27:14	DROP TABLE IF EXISTS `Music`.`Alb...	0 row(s) affected, 1 warning(s): 1051 Unknown table...	0.00029 sec
12	11:27:14	CREATE TABLE IF NOT EXISTS `Mu...	0 row(s) affected	0.019 sec
13	11:27:14	SET SQL_MODE=@OLD_SQL_MODE	0 row(s) affected	0.00025 sec
14	11:27:14	SET FOREIGN_KEY_CHECKS=@OL...	0 row(s) affected	0.00026 sec
15	11:27:14	SET UNIQUE_CHECKS=@OLD_UNI...	0 row(s) affected	0.00028 sec

Query Completed

www.sql-practice.com

We’re going to use a generic online sample RDBMS to practice basic SQL queries



Operators

Operator	Description
=	Equal to
>	Greater than
<	Less than
>=	Greater than or equal to
<=	Less than or equal to
<>	Not equal to

Operator	Description
ALL	TRUE if all of the subquery values meet the condition
AND	TRUE if all the conditions separated by AND is TRUE
ANY	TRUE if any of the subquery values meet the condition
BETWEEN	TRUE if the operand is within the range of comparisons
EXISTS	TRUE if the subquery returns one or more records
IN	TRUE if the operand is equal to one of a list of expressions
LIKE	TRUE if the operand matches a pattern
NOT	Displays a record if the condition(s) is NOT TRUE
OR	TRUE if any of the conditions separated by OR is TRUE
SOME	TRUE if any of the subquery values meet the condition

SQL Functions

- COUNT() gives the number of rows
- AVG() gives the average value
- MAX() gives the maximum value
- MIN() gives the minimum value
- SUM() gives the sum of a range of values

There are lots of other functions that can let us manipulate text, understand more about our data, or perform more advanced operations

We can apply different functions to different types of data - for example, we can't use AVG() to calculate the average of a text type column

Practice

What is the highest price that the hospital pays for an item from the vendor with ID number 4?

Date Functions

- Use YEAR(), MONTH(), or DAY() to extract specific parts of dates
- Use DATEDIFF(unit, date1, date2) to calculate the difference between two dates
  - E.g. DATEDIFF(year, 2013-06-13, 2020-06-13) produces result 7
- Use NOW(), GETDATE(), or SYSDATE() to get the current date (differs based on RDBMS platform)

Practice

Generate a table containing patient_id, the patient's first and last names combined into one variable called patient_name, and the patient's allergies. The table should only include patients with an allergy to penicillin or morphine.

Today's 2nd Most Controversial Slide

JOINS vs. Subqueries

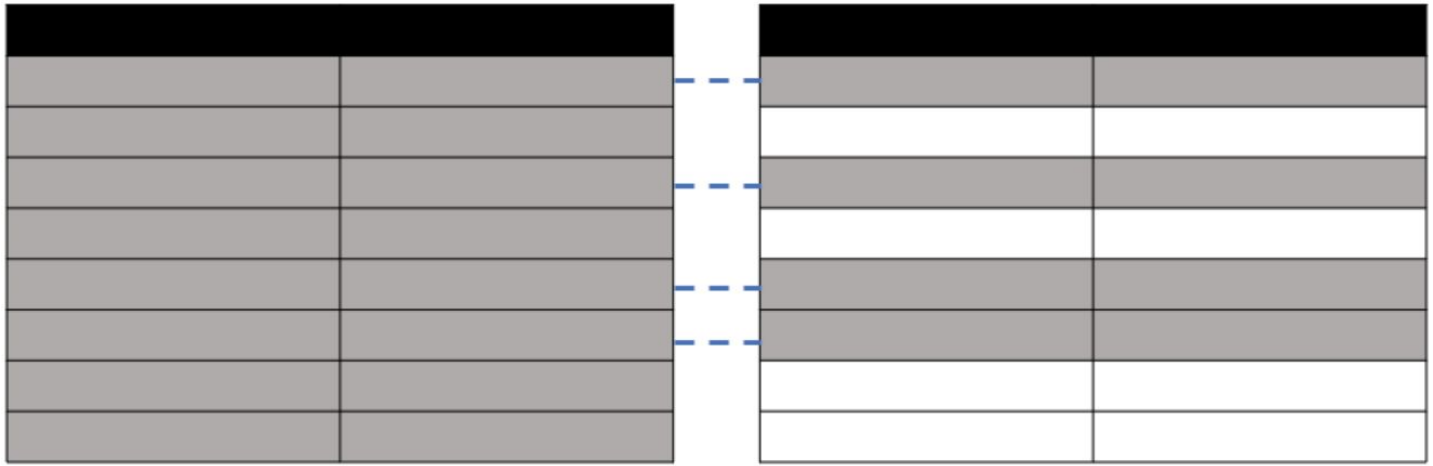
Joins

```
SELECT [columns to return]

FROM [left table]

[JOIN TYPE] [right table]

ON [left table].[field in left table to match] = [right table].[field
in right table to match]
```

Left Join

Left Join

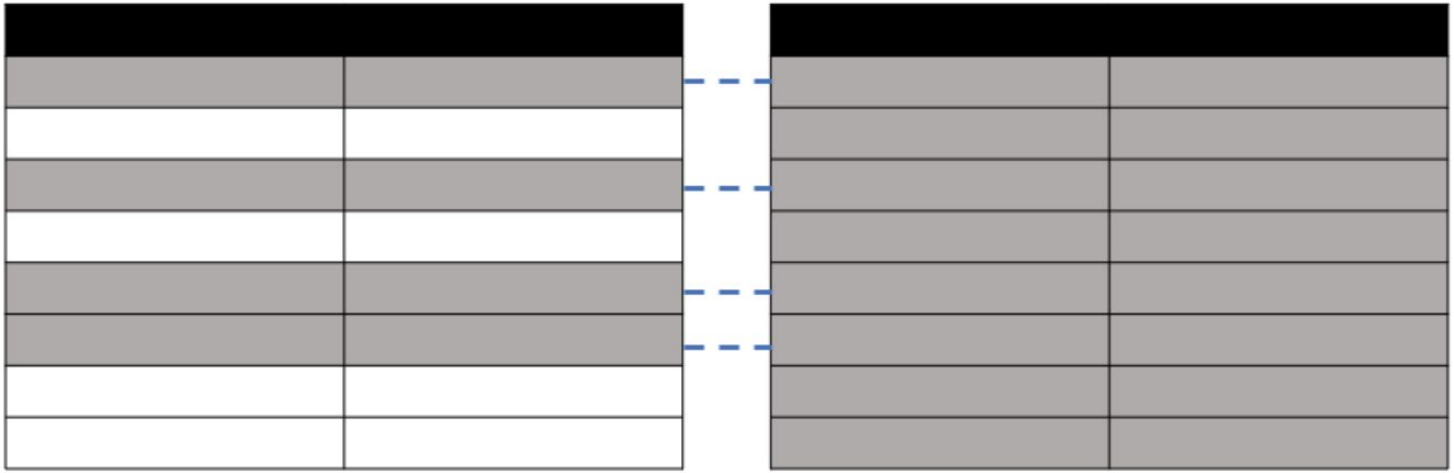
All rows from the “left table”, and only rows from the
“right table” with matching values in the specified fields



Right Join

Right Join

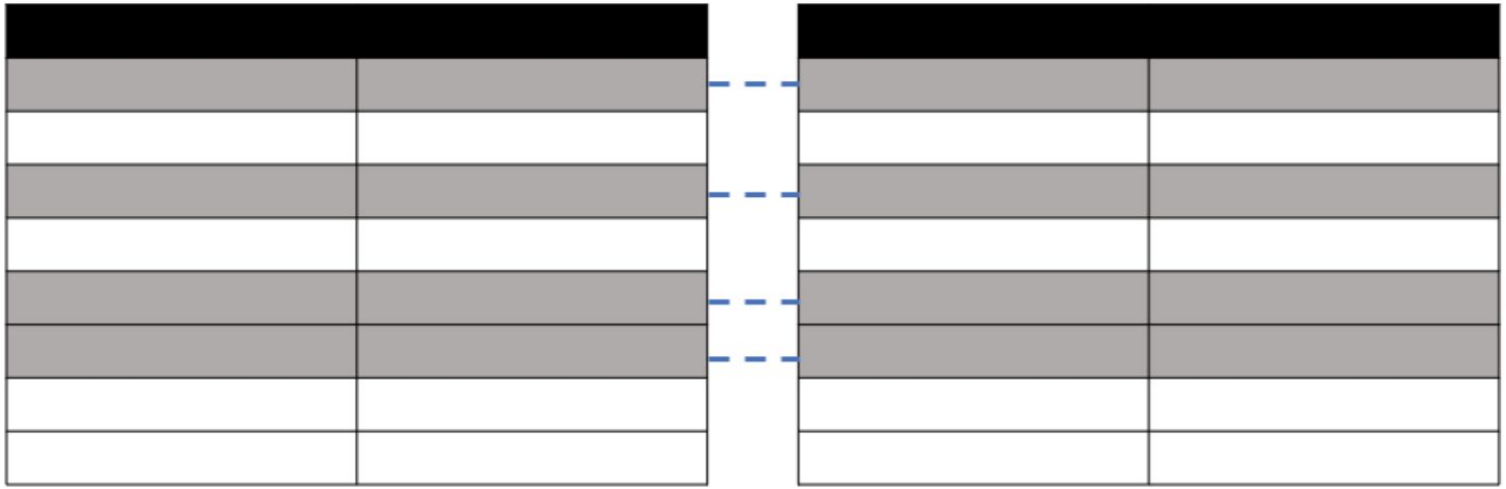
All rows from the “right table”, and only rows from the “left table” with matching values in the specified fields



Inner Join

Inner Join

Only rows from the “right table” and “left table” where values in the specified fields have matches in both tables



Right Join

```
SELECT
  patients.patient_id,
  patients.first_name,
  patients.last_name,
  admissions.room,
  admissions.bed
FROM patients
  RIGHT JOIN admissions ON patients.patient_id = admissions.patient_id;
```

Practice

All patients can access their medical documents on the hospital’s website site. To access their documents, patients are given a temporary password. Show the patient_id and generate a password for each patient.

The password must be the following, in order:

1. patient_id
2. the numerical length of patient's last_name
3. year of patient's birth_date

Practice

We are looking for a specific patient. Pull all columns from the patient table for the patient who matches the following criteria:

- First_name contains an ‘ian’
- Identifies their gender as 'F'
- Born in April, May, or November
- Their weight is between 80kg and 100kg
- Allergies field was unfilled in their patient record (NULL)
- They are from the city of Hamilton

Practice

Generate a table that shows the number of male patients from each province as num_patients. The province’s full name (i.e. not the shortened province_id) must be shown in the table.


WAHOO!!

Today's Goal:

Be able to go into a relational database and pull out specific subsets of data using SQL.