SQL 101 – Intro to SQL for Clinical Data

ARCUS Education

Children's Hospital of Philadelphia

Today’s Itinerary

- Overview of clinical data and its complexities
- SQL 101
- What is the ADR?
- Where to learn more

Children’s Hospital of Philadelphia®

Things to keep in mind

The most important things to pay attention too in this lecture (and in any intro course really) is the new "**vocabulary**" & "**concepts**" that are introduced.

- Rather than trying to completely understand every step of the example we run through, or thinking that you need to fully understand all of this training material the first time around.

Children's Hospital of Philadelphia®

What makes Clinical Data so Complex?

Short Answer –

The Inherent complexity in the field & shear amount of information captured/recorded.

Long Answer -

Data will "fragmented" in some way when it is stored in your database; *order to minimize the "cost" of storing and accessing your data.*

- This "fragmentation" of data means that you will usually have to do some work get exactly what you want (this is where sql will come in!).

Children's Hospital of Philadelphia®

What's with the ”Fragmentation”?

One “**Patient**” can have multiple “**Encounters**” (even in a single day)

During each encounter:

They can be assigned multiple “**Diagnoses**” …

They can have multiple “**Orders**” placed for them …

They can undergo multiple “**Labs**” & “**Procedures**” …

Each lab or procedure can have multiple “**Components**” …

They can have multiple “**Medication**” prescribed or administered …

Each medication can have “**Dosage**” and “**Formulation**” information …

… You get the idea, no way you can easily store all of that data information in one “place”

Children's Hospital of Philadelphia®

Okay so seems like data ”Fragmentation” is necessary …

but then how do we work with it?

“**SQL**” (**S**tructured **Q**uery **L**anguage) is the name of the programing language that we (humans) use to interact with “**Relational Databases**” (computers).

You can think of **SQL** as the super secret code that people can use to explicitly “*ask questions*” that you would like your **Database** to answer.

Before we get our first look at some **SQL** code, let’s start out by considering what **Clinical Data** is like…

Children’s Hospital of Philadelphia®

Normalization

In Database lingo, this "fragmentation" of data that we have been talking about actually has a specific name, "**Normalization**".

- **Normalization** means that the diagnoses associated with an encounter are stored in one "place", and data about the actual encounter itself is stored in another "location" (i.e. data is stored in different places based on its "context" or "domain").

- **Normalization** avoids repetition of the data (and makes it easier to manage/update).

Sources Like the **ADR** and **CDW** have been created with the goals of reducing fragmentation (when compared to **Clarity**), but no matter the data source a certain amount of fragmentation is unavoidable.

Children's Hospital of Philadelphia®

A Non-Normalized Dataset

224 cells, lots of repetition

PAT_MRN	PAT_DOB	RACE	ETHNICITY	SEX	ICD_10	GENERIC_NM
356	11/26/00	5	2	1	C84.4	Ketorolac Tromethamine Inj 15 MG/ML
356	11/26/00	5	2	1	C84.4	Ibuprofen Susp 100 MG/5ML
356	11/26/00	5	2	1	C84.4	Ibuprofen Tab 600 MG
356	11/26/00	5	2	1	C84.A5	Ketorolac Tromethamine Inj 15 MG/ML
356	11/26/00	5	2	1	C84.A5	Ibuprofen Susp 100 MG/5ML
356	11/26/00	5	2	1	C84.A5	Ibuprofen Tab 600 MG
564	12/13/13	5	2	0	S83.5	Ibuprofen tablet
564	12/13/13	5	2	0	S83.5	Ibuprofen
564	12/13/13	5	2	0	S83.5	Ketorolac Tromethamine Inj 30 MG/ML
564	12/13/13	5	2	0	S83.5	Ibuprofen Tab 400 MG
564	12/13/13	5	2	0	S83.1	Ibuprofen tablet
564	12/13/13	5	2	0	S83.1	Ibuprofen
564	12/13/13	5	2	0	S83.1	Ketorolac Tromethamine Inj 30 MG/ML
564	12/13/13	5	2	0	S83.1	Ibuprofen Tab 400 MG
675	3/3/13	3	2	1	G44.001	Ibuprofen suspension
675	3/3/13	3	2	1	G44.001	Ibuprofen Tab 200 MG
844	11/28/04	7	2	0	D70.3	Ibuprofen Tab 400 MG
844	11/28/04	7	2	0	D70.3	Ibuprofen Susp 100 MG/5ML
844	11/28/04	7	2	0	D70.3	Ketorolac Tromethamine Inj 30 MG/ML
844	11/28/04	7	2	0	D70.3	Ibuprofen Tab 800 MG
923	4/5/07	5	2	0	S83.5	Ibuprofen tablet
923	4/5/07	5	2	0	S83.5	Ibuprofen Tab 600 MG
923	4/5/07	5	2	0	S83.5	Ketorolac Tromethamine Inj 30 MG/ML
923	4/5/07	5	2	0	S83.5	Ibuprofen Susp 100 MG/5ML
923	4/5/07	5	2	0	S83.5	Ibuprofen Tab 400 MG
923	4/5/07	5	2	0	S83.5	Ibuprofen Tab 200 MG
942	9/1/06	NA	NA	0	M60.029	Ibuprofen Susp 100 MG/5ML
942	9/1/06	NA	NA	0	M60.029	Ibuprofen tablet
942	9/1/06	NA	NA	0	M60.002	Ibuprofen Susp 100 MG/5ML
942	9/1/06	NA	NA	0	M60.002	Ibuprofen tablet
942	9/1/06	NA	NA	0	B01	Ibuprofen Susp 100 MG/5ML
942	9/1/06	NA	NA	0	B01	Ibuprofen tablet

Children's Hospital of Philadelphia®

Same Dataset, but Normalized

PAT_MRN	PAT_DOB	RACE	ETHNICITY	SEX
923	4/5/07	5	2	0
942	9/1/06	NA	NA	0
356	11/26/00	5	2	1
844	11/28/04	7	2	0
675	3/3/13	3	2	1
564	12/13/13	5	2	0

PAT_MRN	ICD_10
923	S83.5
942	M60.029
942	M60.002
942	B01
356	C84.4
356	C84.A5
844	D70.3
675	G44.001
564	S83.5
564	S83.1

92 total cells, minimal repetition

"Tables" can be "joined" (i.e. connected or linked) on ***pat_mrn***

PAT_MRN	GENERIC_NM
923	Ibuprofen Susp 100 MG/5ML
923	Ibuprofen tablet
923	Ibuprofen Tab 600 MG
923	Ketorolac Tromethamine Inj 30 MG/ML
923	Ibuprofen Tab 200 MG
923	Ibuprofen Tab 400 MG
942	Ibuprofen Susp 100 MG/5ML
942	Ibuprofen tablet
356	Ketorolac Tromethamine Inj 15 MG/ML
356	Ibuprofen Susp 100 MG/5ML
356	Ibuprofen Tab 600 MG
844	Ketorolac Tromethamine Inj 30 MG/ML
844	Ibuprofen Tab 800 MG
844	Ibuprofen Tab 400 MG
844	Ibuprofen Susp 100 MG/5ML
675	Ibuprofen suspension
675	Ibuprofen Tab 200 MG
564	Ketorolac Tromethamine Inj 30 MG/ML
564	Ibuprofen Tab 400 MG
564	Ibuprofen tablet
564	Ibuprofen

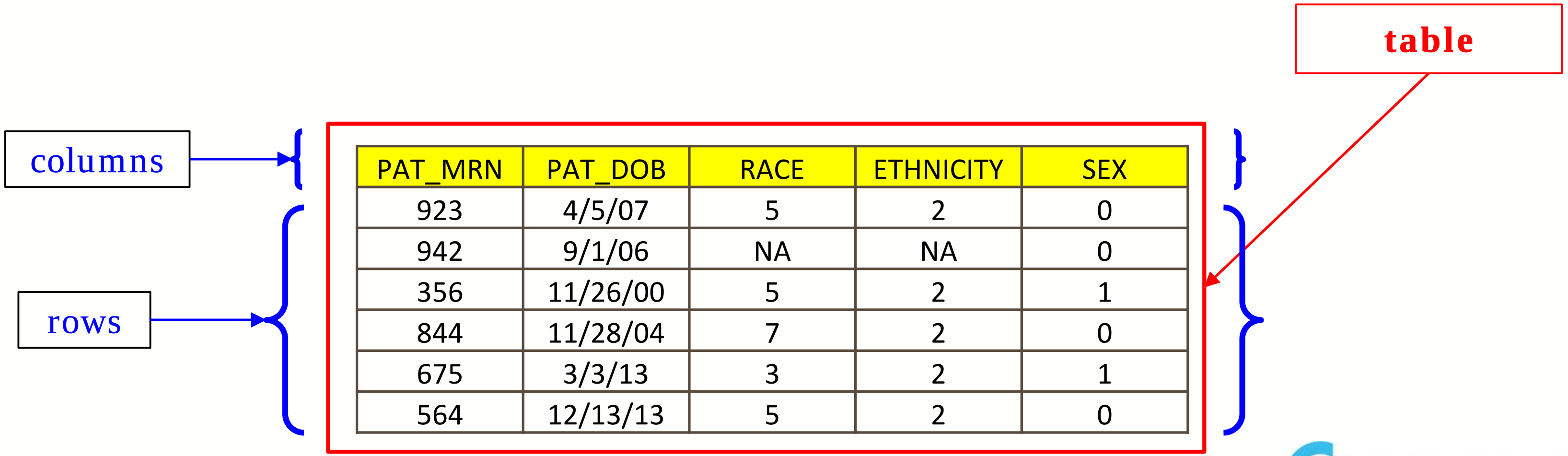
Children's Hospital of Philadelphia®

How is Data actually Formatted/ Stored ?

SQL databases are made up of **tables**, and each table has **columns** and **rows**.

Columns are fields like: “mrn”, “year”, “dx_id”, “race”, “BMI”, “payor_name”, etc.

Rows are the actual observations or data points that make up each column of the table.



PAT_MRN	PAT_DOB	RACE	ETHNICITY	SEX
923	4/5/07	5	2	0
942	9/1/06	NA	NA	0
356	11/26/00	5	2	1
844	11/28/04	7	2	0
675	3/3/13	3	2	1
564	12/13/13	5	2	0

SQL Query Structure

You're going to have to answer three questions to construct the most basic SQL query:

- What's the name of the **table(s)** where the data is stored?
- What **column(s)** (*a.k.a field(s)*) from the table do you want to look at?
    - e.g. study id, patient id, year when measure was taken, score or value, etc.
    - If you want everything, you'll use an asterisk to indicate that.
- What **filtering condition(s)** do you want to apply to your view of the data (*this will determine what **rows** your view of the data contains*)?
    - e.g. Only males, only the rows with non-empty values for a certain field, only the rows where the year is 2017, etc.
    - Though if you want to pull every record from your data then you don't actually need to provide a filtering condition.

Children's Hospital of Philadelphia®

SQL "Flavors"

All SQL "flavors" are similar, *in that they all follow the same ANSI SQL syntax*.

However there usually a few small differences between each "flavor". The most commonly difference being that each SQL flavors will have own "custom functions" that can be used to transform input data (i.e. to replace parts of a text string, find the max value among several columns, etc.).

Some Popular SQL Flavors:

● SQLite	(open source)	← We will be using this today!
● MySQL	(open source)	
● PostgreSQL	(open source)	
● Oracle	(proprietary)	← Used by Clarity (export of Epic Data)
● Netezza	(proprietary)	← Used by CDW (Chop Data Warehouse)
● BigQuery	(proprietary)	← Used by ARCUS (Chop ADR)

Children's Hospital of Philadelphia®

Basic SQL Query Syntax

You put the most basic query together like this:

SELECT _ FROM_ WHERE_ ;

While SQL keywords can be any case, sometimes we put these words in all caps to make them stand out.

```
-- Example Query #1
SELECT
        field_i_want       ,
        other_field_i_want
FROM table_im_interested_in
WHERE year > 2017
;


-- Example Query #2
SELECT *
FROM nutritional_value
WHERE LOWER(food) IN ('apple', 'banana', 'grapefruit')

;
```

Useful Extra… the “LIMIT” filter

```
SELECT *
FROMtablename
LIMIT    20
;
```

Gives you a preview of what the table looks like by limiting the result set to the first 20 rows returned.

Lets Download a Fake Clinical Dataset that we Can Play with

Lets dive right into writing some SQL queries in **SQLite** against a "Fake" Clinical Dataset (*links for Data and SQLite App below*).

Download fake data from sharefile:

https://chop.sharefile.com/d-s57101c68c9148cc9

DB Browser for SQLite:

https://sqlitebrowser.org/dl/

Basic SQL Query - Try it With Me!

Let’s look at the first 20 rows of a few different tables!

```
SELECT *
FROM patients
LIMIT   20
;
```

Children’s Hospital of Philadelphia®

Basic SQL Query - Adding Comments

We can add “comments” to our queries to help us remember what the various fields are.

```
SELECT
        id,             --  unique patient identifier
        birthdate,      --  patient date of birth
        race,           --  patient race
        ethnicity,      --  patient ethnicity
        deathdate       --  patient date of death
FROM patients
LIMIT   10
;
```

Basic SQL Query - Filtering Results

Note the single quotes around ‘Boston’, that’s important!

Also, check to see what happens if you type a lower case **B (i.e. ‘boston’)

```
SELECT
      id,                -- project specific patient identifier
      birthdate,         -- patient date of birth as year
      race,              -- patient race
      ethnicity          -- patient ethnicity
FROM patients
WHERE
      patients.city         = 'Boston'
```

Children’s Hospital of Philadelphia®

Translating your “Question” into a Query

Question #1:

Write a query that selects ***race*** and ***ethnicity*** for only those **patients** that live in ***Boston***.

Question #2:

Select all patient demographic data (everything in the “patients” table) for only those **patients** that live in ‘**Boston**’ or ‘**Cambridge**’ and whose **year of birth** is greater than 2000.

Question Solutions

```
--  Challenge Query #1 Solution.
SELECT race,ethnicity          FROM patients
WHERE lower(city) =          ' boston  '

--    Challenge Query #2 Solution.
SELECT *
FROM patients
WHERE
        (city =       'Boston'      OR city = 'Cambridge        ' )
        AND birthdate > '2000         '
```

Children's Hospital of Philadelphia®

“DISTINCT”

What if you wanted to know how many unique zip codes (or records) are included in your result set?

```
SELECT DISTINCT
        zip
FROM patients
WHERE
      (city = 'Boston' OR city = 'Cambridge')
      AND birthdate > '2000';
```

“GROUP BY”

The **GROUP BY** clause is used any time that you want summarize or aggregate information; as it tells the database the “level” that you want your data “collapsed too”.

For example you can use this syntax, in connection with the **COUNT()** function, to find out how many patients have home addresses in each distinct zip code:

```
SELECT
        zip
        , COUNT(*) as count
FROM patients
where
        (city = 'Boston'            OR city =
'Cambridge')
        AND birthdate >  '2000'
GROUP BY
        zip
```

Children's Hospital of Philadelphia®

Using Aliases

In SQL, you can use "Aliasing" to provide convenient **"nick-names"** for **tables** or **columns**.

This can be helpful if you either don't want to type the full name of a table every time you reference it, and/or if there are columns that you want to rename (e.g. if the original name is non-descript).

```
SELECT
    p.id as patient_id
FROM patients as p
ORDER BY
    patient_id DESC
```

Pause for Questions...

Next, we'll look at a real Clinical Data source that researchers at CHOP can gain access too called the ADR (**A**rcus **D**ata **R**epository).

But before we do, now seems like a good point to pause for questions:

Any questions on what we've covered so far?

Anything your still curious about at this point regarding SQL syntax ?

Children's Hospital of Philadelphia®

The Arcus Data Repository

The ADR is a simplified database of CHOP patient data, with many commonly-requested fields, arranged in a small number of tables.

It is not a complete copy of Epic data!

You can explore more via a SQL front end.

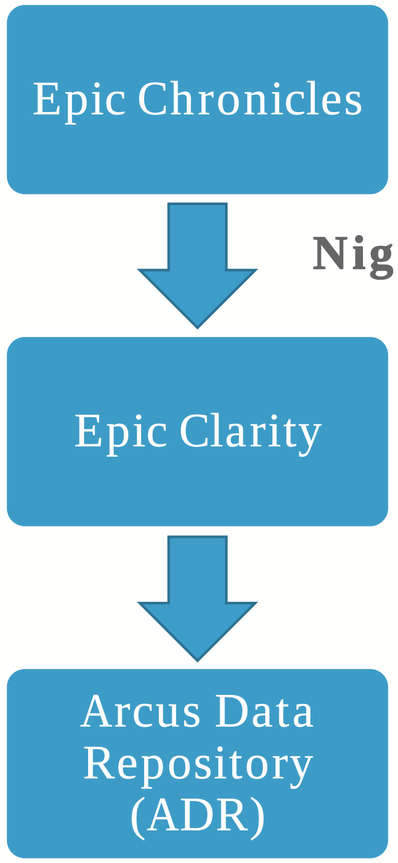
You'll be looking at the de-identified version.

- so, non human subjects research is possible…

Children's Hospital of Philadelphia®

High Level ARCUS ETL Pipeline

~100,000 Data Elements
Non-Normalized
Document Storage Model



Used for clinical, real—time applications (Hyperspace, etc.)

Nightly data transfer

Normalized
12,000+ **tables**
125k **columns**

1 *day behind live EPIC data.*

Normalized & Simplified
~30 **tables**
2k **columns**

In-sync with Clarity Daily Refresh

Children's Hospital of Philadelphia®

So, Who’s In the ADR?

The **ADR** (**A**rcus **D**ata **Repository**) contains Patients who:

- Are Considered “Valid”:
  - Patients must have at least one “face-to-face” encounter in Clarity/Epic.
  - Test patients are excluded.

Note: ARCUS also keeps track of which patients are excluded from the ADR for auditing purposes (if your ever curious and want to reach out to them about validation).

So, What’s In the ADR?

Short Answer –

- Patient (Demographics about patient)
- Encounter (Including ADT events)
- Diagnoses (At encounter and in problem list)
- Medication (Orders and inpatient administration)
- Procedures (Including results, labs)
- Immunizations
- Coverage
- Smart Data Elements**
- Flowsheets **
- Notes and note annotations **

**Not yet available in the deidentified/coded schema’s

Children’s Hospital of Philadelphia®

So, What's In the ADR?

Long Answer –

We can actually take a look at this ourselves (Provided we have the necessary CITI training completed) from the ARCUS application web portal (only available from CHOP network):

Lets check out the site! https://app.arcus.chop.edu

Follow Along with me:

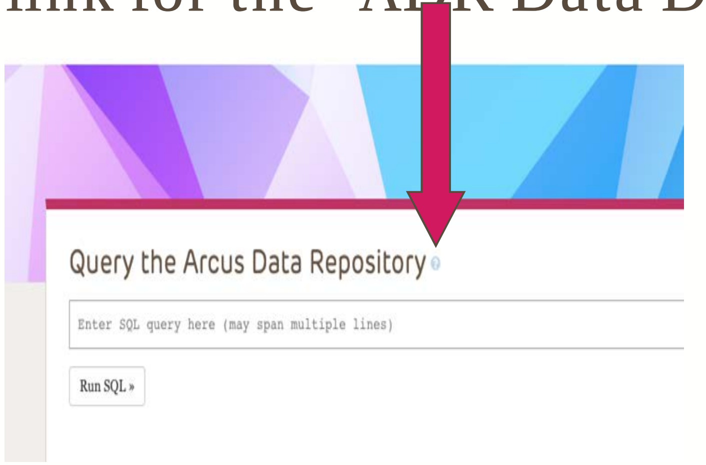
- Log in (button in upper right corner of the page)
- Sign the terms of use
- Go to the "Data Repository Query" tab

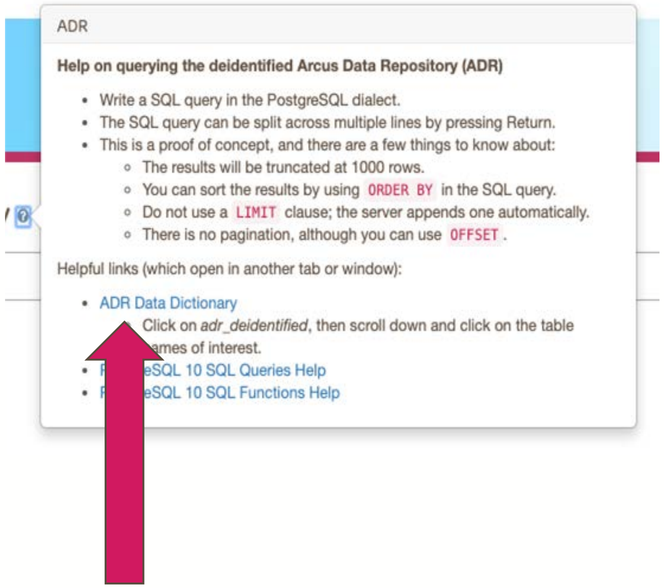
Children's Hospital of Philadelphia®

So, What’s In the ADR?

You can preview the data schema (the “contents of the database”) by following the “question mark” path in the same tab.

Hover over that question mark icon, then click on the link for the “ADR Data Dictionary”.


Query the Arcus Data Repository
Enter SQL query here (may span multiple lines)
Run SQL »



ADR

Help on querying the deidentified Arcus Data Repository (ADR)

- Write a SQL query in the PostgreSQL dialect.
- The SQL query can be split across multiple lines by pressing Return.
- This is a proof of concept, and there are a few things to know about:
  - The results will be truncated at 1000 rows.
  - You can sort the results by using `ORDER BY` in the SQL query.
  - Do not use a `LIMIT` clause; the server appends one automatically.
  - There is no pagination, although you can use `OFFSET`.

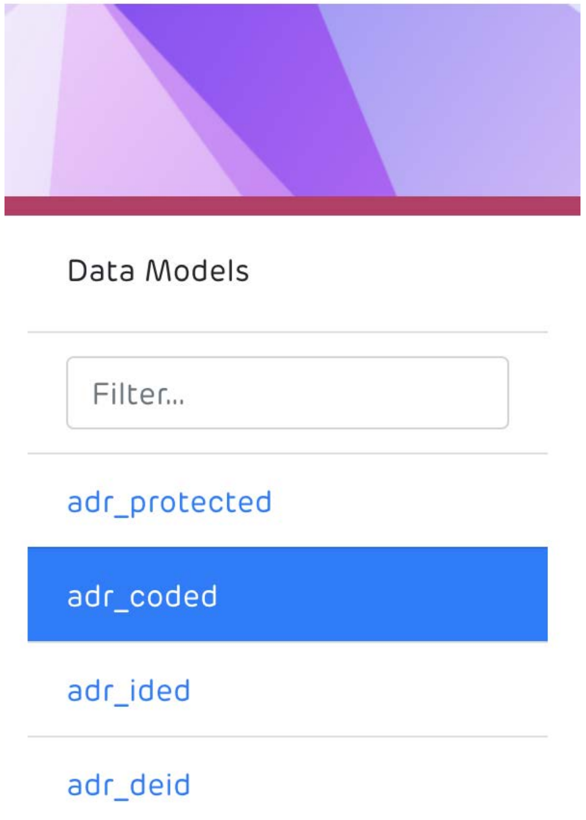
Helpful links (which open in another tab or window):

- ADR Data Dictionary
  - Click on *adr_deidentified*, then scroll down and click on the table names of interest.
- PostgreSQL 10 SQL Queries Help
- PostgreSQL 10 SQL Functions Help

So, What's In the ADR?

Once you have the "ADR Data Dictionary" application open, go to the section of the page labeled "Data Models" and choose the option titled "**adr_coded**"

You can then look at what's contained in various tables available in that "data model" (by exploring the section of the page titled "tables" and clicking on the name of each table that is listed).



Data Models

Filter...

adr_protected

adr_coded

adr_ided

adr_deid

Example query in the ADR

```
SELECT
        Language
        , COUNT(*)      AS  patient_count
FROM patient
WHERE
        (state = 'New York'          OR state = 'New Jersey')
        AND language != 'English'
GROUP BY
        language
HAVING
        patient_count       > 10
ORDER BY
        patient_count       DESC
;
```

Children's Hospital of Philadelphia®

How to Learn More!

- Khan Academy has a nice Videos intro course on SQL.

- w3schools has some great “read along” learning modules for SQL.

- SQLite & BigQuery also have their own documentation sites.
  - Note: These last 2 resources may be harder to read for newer users so its okay to feel intimidated, however there is a lot of great info in there!

Children's Hospital of Philadelphia®

Pause for Questions...

Plug for Part 2 of this Course

Please come back for **SQL 102** !

In this part 2 lecture we will expand on the topics covered in this lesson, and we will introduce the concept of SQL **joins** (*i.e. the SQL syntax used to "merge" or "link" multiple tables together*).

See a sneak preview of SQL 102 on the next 3 slides ...

...And tune into the next SQL 102 session for more info!

Joins

Because of Normalization, most queries require something more complex than only referencing a single table.

You will often need to **combine data from two or more tables**.

This is where SQL **JOIN(s)** comes into play.

There are two basic things you need for a successful join:

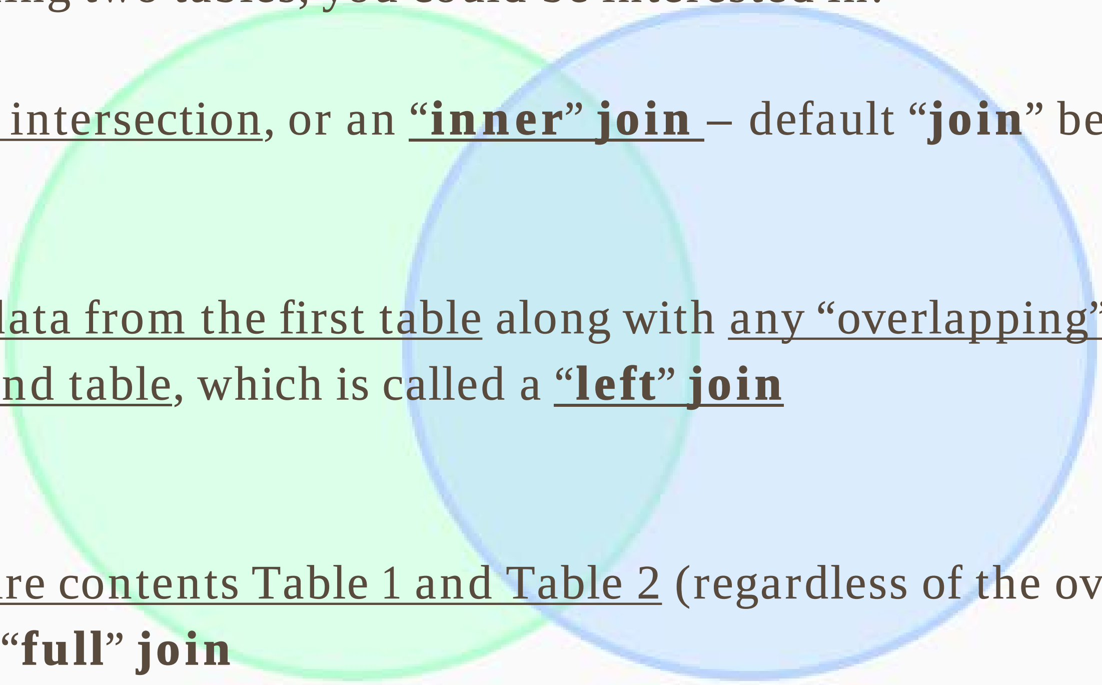
What Join Criteria –
What Columns/Keys that can be used to "connect" (i.e. "join") your 2 tables?

How to Apply that Join Criteria –
What part of the "Venn Diagram" of the overlapping are you interested in?

- *More on this in the next slide.*

Children's Hospital of Philadelphia®

Join Criteria



Its helpful to imagine a join as one of the combinations of a “Venn diagram”.

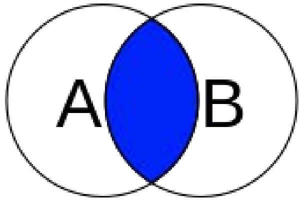
If you’re joining two tables, you could be interested in:

- Just the intersection, or an **“inner” join** – default “**join**” behavior

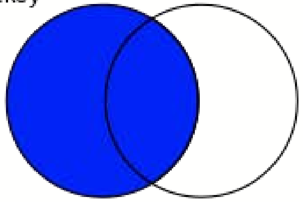
- All the data from the first table along with any “overlapping” data from the second table, which is called a **“left” join**

- The entire contents Table 1 and Table 2 (regardless of the overlap) is called a “**full**” **join**

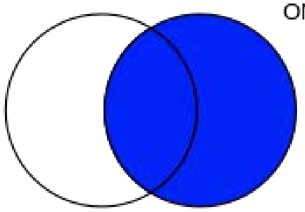
```
SELECT <fields>
FROM TableA A
INNER JOIN TableB B
ON A.key = B.key
```



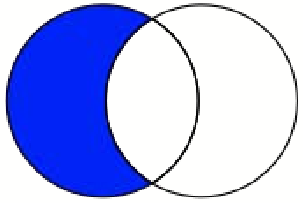
```
SELECT <fields>
FROM TableA A
LEFT JOIN TableB B
ON A.key = B.key
```



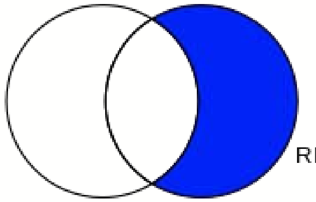
```
SELECT <fields>
FROM TableA A
RIGHT JOIN TableB B
ON A.key = B.key
```



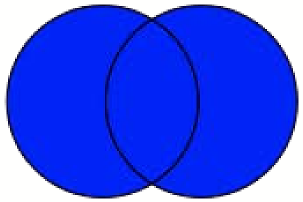
SQL JOINS



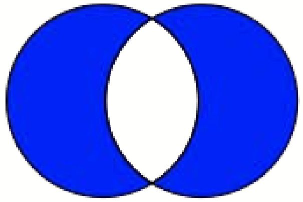
```
SELECT <fields>
FROM TableA A
LEFT JOIN TableB B
ON A.key = B.key
WHERE B.key IS NULL
```



```
SELECT <fields>
FROM TableA A
RIGHT JOIN TableB B
ON A.key = B.key
WHERE a.key IS NULL
```



```
SELECT <fields>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.key = B.key
```



```
SELECT <fields>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.key = B.key
WHERE A.key IS NULL
OR B.key iIS NULL
```

Children's Hospital of Philadelphia®