SQL for the Absolute Beginner

Kimberly Smith, PhD, MT(ASCP)
Assistant Professor
School of Biomedical Informatics
The University of Texas Health Science Center at Houston
Kimberly.A.Smith@uth.tmc.edu
6/23/2019
About the Tutorial Database

• All records are fake/synthetic data
• Intended for teaching SQL concepts
  – Not designed in the same way as a real EHR system
  – Not medically correct
• Installed on your own computer – not shared (You can’t hurt anything!)
Objectives

• Use DESCRIBE, SELECT, and WHERE to perform simple queries
• Describe the difference between * and wildcard characters and correctly use them in a SQL query
• Use pattern matching with LIKE, NOT LIKE, and =, and use Boolean operators AND, OR, and NOT
• Perform basic mathematical functions with SQL, such as averages and counts
• Use JOIN to combine data from two tables
What is SQL?

• Structured Query Language
• Pronounced either “SEQUEL” or Es-Que-El
• A very common language used to interact with databases
Using SELECT to Retrieve Data
1. Double-click the Assignment schema

2. Type queries here...

3. Then highlight your query and click the lightning bolt with cursor to run just that query

4. List of patients that match your query will appear here

5. Status messages from the system, such as number of records, error messages, or successful completion, will appear here
Asking Questions

• When data are stored in a database, how do we extract information from that data?
  – How many patients have been diagnosed with disease X in timeframe Y?
  – How many patients does Dr. Jones have?
  – What is the average hemoglobin A1C for Dr. Jones’ patients, compared to all all patients?
You Try It!

Type:

```
SELECT LastName, FirstName
FROM Patients;
```

To run the query:
1. Click on somewhere on your query
2. Then click the lighting bolt icon with the cursor mark
Syntax:

```sql
SELECT column_name, column_name
FROM table_name;
```
Simple SELECT Example, explained

• FROM statement identifies where the data are
• You must understand the structure of the table
• The query must end with a semicolon
Summary of SELECT Statements

• A SELECT statement contains a complete description of a set of data that you want to obtain from a database.

• This includes the following:
  – What tables contain the data
  – How data from different sources are related
  – Which fields or calculations will produce the data
  – Criteria that data must match to be included
  – Whether and how to sort the results
The SELECT Statement

• The SELECT statement is probably the most commonly used in SQL. It simply retrieves data from the database.
  – **Selects** *what*
  – **From** *specific locations in the database* (such as tables)
  – **Where** certain criteria are met (e.g. filters)
Cautions, Tips, and Hints
CAUTION!

• SQL assumes you know what you are doing
• If you issue a command to delete the entire database, it will not ask, “Are you sure?”
• Production systems do have “rollback” functions, but in general, assume there is no “undo”
• This system is a training one, so you can’t hurt anything!
A Note on SQL Dialects

• Many flavors, dialects, and proprietary versions of Structured Query Languages
• Microsoft Access database uses Jet
• Microsoft SQL Server database uses Transact-SQL (T-SQL)
• Oracle database has its own
• Cerner EHRs use a SQL-like language called Cerner Command Language (CCL)
• This lecture uses MySQL conventions
Tips and Hints

• Use **single quote marks** around groups of letters (these are called *strings*)
• Do not use **double quote marks**
• Copying and pasting from Word into MySQL can cause problems because Word uses a different character called a smart quote – best to just type in the quote marks directly in MySQL
• Capitalization of tables **must match exactly** how the table is named in the database: Patients is different than patients.
Tips and Hints, 2

• Be patient with yourself!
• Set aside time to practice
• Expect to be frustrated at points – you are learning a language, and this takes time
• Lots of resources available on the web – search using “mysql how to ...”
Attributes or Properties

PATIENT
Medical record #
Name
Address
Table with Data

• Tables have two parts
  1. Heading (or columns)
  2. Body (or rows)
Table Attributes or Properties

- The name of the table
- The primary identifier (key) for this table
- Indicates required data
- Identifiers from Location, Provider, and Care Site tables
A Birds-Eye View of Several Tables in One Database

An Analogy – Part 1

• Think of a house being built
  – Blueprints and floor plans
  – Foundation and exterior walls
  – Rooms with specific functions
  – Furniture, appliances, plumbing

https://commons.wikimedia.org/wiki/File:Sample_Floorplan.jpg

https://commons.wikimedia.org/wiki/File:Kitchen_interior_design.jpg
An Analogy – Part 2

• **Database** – like the walls and foundation of a house
• **Tables** – like rooms for the data
• **Entries** (rows) in tables – each row is a specific patient, doctor, medication, etc.
• **Schema or Entity Relationship Diagrams (ERDs)** are like blueprints and floor plans – visualizations of the database with all the tables and their relationships
• You will see “schema” and “database” used interchangeably
Using DESCRIBE
The DESCRIBE Command

• Essential to understand where data are in the database

• Example:
  – Where is the patient name? How many characters is it? What data type is it?
  – Where are the lab results? Where are the medications? Where are the physicians? And so on

• DESCRIBE \textit{TABLENAME} gives you basic information
Explore the Patients table structure
Let’s look at a table!
You Try It!

Run this command:

```sql
DESCRIBE Patients;
```

How does this compare to an Excel spreadsheet?
Selecting only specific columns
1. Select the last name, first name, and occupation for all patients

2. What is the occupation of the 14th patient on the list?
SELECT LastName, FirstName, Occupation
FROM Patients;

What is the occupation of the 14th patient on the list?
SELECT LastName, FirstName, Occupation
FROM Patients;
Trader
Databases
Storing Data

- Let’s consider an Excel file – what are some drawbacks to storing data in this type of file?

<table>
<thead>
<tr>
<th>PatientName</th>
<th>PatientPhone</th>
<th>DoctorName</th>
<th>DoctorPhone</th>
</tr>
</thead>
<tbody>
<tr>
<td>Doe, John</td>
<td>123-456-7890</td>
<td>Smith, K</td>
<td>512-500-3920</td>
</tr>
<tr>
<td>Smith, Mary</td>
<td>123-654-0987</td>
<td>Jones, T</td>
<td>512-500-3921</td>
</tr>
<tr>
<td>Jones, Jill</td>
<td>123-321-2345</td>
<td>Smith, K</td>
<td>512-500-3920</td>
</tr>
<tr>
<td>Johnson, Scott</td>
<td>123-456-6789</td>
<td>Jones, T</td>
<td>512-500-3921</td>
</tr>
<tr>
<td>Baker, Tom</td>
<td>123-345-5678</td>
<td>Garcia, C</td>
<td>512-500-3923</td>
</tr>
</tbody>
</table>
Problems with Files

• Data are often duplicated (*data integrity*)
• Data can be inconsistent (*200 mg/dl glucose vs. glucose, 200 mg/dl*)
• Data can be incompletely removed
• Partial data can be entered
• Difficult to represent data in users’ perspectives
Relational Database

• *Relational database*: An organized set of related data

• Databases are made up of *tables* – each table has one theme, such as Patients, Physicians, or Diagnoses

• Widely used in health IT systems, such as EHRs, laboratory systems, radiology systems, and so on
Two Main Functions of SQL

1. Working with the *content* of the database
   (Data Manipulation Language / DML)

Used to insert, select, update, and delete records in the database. Include:

- **SELECT** - Retrieves data from the database
- **INSERT** - Inserts new data into the database
- **UPDATE** - Updates existing data in the database
- **DELETE** - Deletes existing data from the database
Two Main Functions of SQL (continued)

2. Modifying the *structure* of the database itself (Data Definition Language / DDL commands)

Examples of DDL commands:

- **CREATE DATABASE** - Creates a new database
- **ALTER DATABASE** - Modifies the database
- **DROP DATABASE** - Drops (deletes) a database
- **CREATE TABLE** - Creates a new table
- **ALTER TABLE** - Modifies the table structure
- **DROP TABLE** - Drops (deletes) a table
Selecting all the data for every record
Selecting Everything

• If you want all the fields for each patient, you could type every column name
• OR, you could just use a single * instead of listing the columns
You Try It!

1. Select all columns for all patients in the Patients table
2. How many patients are on the list?
3. What is the first name of the first patient on the list?
SELECT * 
FROM Patients;

2. How many patients are on the list?

3. What is the first name of the first patient on the list?
1. SELECT * 
   FROM Patients;

2. 790

3. What is the first name of the first patient on the list?
1. SELECT * 
   FROM Patients;

2. 790

3. Debbie
Simple SELECT Example, explained

- **SELECT * FROM Patients;**
- What this means:
  - * is a shortcut that will retrieve all columns of data (faster than typing all the column names individually)
Adding Selection Criteria (Filters) with WHERE
Filtering the Selection

- But what if you decide you don’t want all the patient data from the Patients table? You only want patients named Smith.
- You add a “WHERE” clause to your query

Patients Table

<table>
<thead>
<tr>
<th>ID</th>
<th>LastName</th>
<th>FirstName</th>
<th>StreetAddress</th>
<th>City</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Doe</td>
<td>John</td>
<td>123 Main Street</td>
<td>Houston</td>
</tr>
<tr>
<td>2</td>
<td>Jones</td>
<td>Mary</td>
<td>425 Allison Drive</td>
<td>Austin</td>
</tr>
<tr>
<td>3</td>
<td>Smith</td>
<td>Erica</td>
<td>327 Maple Drive</td>
<td>Austin</td>
</tr>
</tbody>
</table>
The WHERE Clause, 1

- Syntax:

```sql
SELECT column_name, column_name
FROM table_name
WHERE column_name LIKE 'value';
```

<table>
<thead>
<tr>
<th>ID</th>
<th>LastName</th>
<th>FirstName</th>
<th>StreetAddress</th>
<th>City</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Doe</td>
<td>John</td>
<td>123 Main Street</td>
<td>Houston</td>
</tr>
<tr>
<td>2</td>
<td>Jones</td>
<td>Mary</td>
<td>425 Allison Drive</td>
<td>Austin</td>
</tr>
<tr>
<td>3</td>
<td>Smith</td>
<td>Erica</td>
<td>327 Maple Drive</td>
<td>Austin</td>
</tr>
</tbody>
</table>
SELECT *  
FROM Patients  
WHERE LastName LIKE 'smith';

- **LastName**: the column where the data item is stored  
- **LIKE ‘smith’**: is used to match for the character string smith
You Try It!

1. What query would give you the first and last name of the patient with medical record number 802?

2. What is the last and first name of the patient with medical record number 802?
Answer

SELECT * 
FROM Patients 
WHERE MRNO LIKE '802';

2. What is the last and first name of the patient with medical record number 802?
SELECT LastName, FirstName
FROM Patients
WHERE MRNO LIKE '802';

2. What is the last and first name of the patient with medical record number 802?
Answer

SELECT * 
FROM Patients 
WHERE MRNO LIKE '802';

Santos, Brenda
You Try It!

1. What query would give you the number of male patients in the Patients table?

2. How many male patients are in the Patients table?
SELECT * FROM Patients WHERE Gender LIKE 'M';

2. How many male patients are there in the Patients table?
SELECT * FROM Patients
WHERE Gender LIKE 'M';
390
Wildcards
Wildcards

• A wildcard character can be substituted for any other character(s) in a string.
• % = for 0 to many characters
• _ (underscore) = for a single character
Wildcards

• ‘kimberly’ will list only patients with the name of Kimberly (exact match, no wildcard)

• ‘kimberl%' will list patients with the name Kimberly, Kimberley, and Kimberlee

• ‘%smith%' will list patients with smith anywhere in the last name: Smith, Smithson, Blacksmith, etc.

• ‘196_’ would list all patients with a date of birth from 1960 through 1969, inclusive
You Try It!

1. What query would give you the number of patients born in 1959?

2. How many patients were born in 1959?
SELECT * FROM Patients
WHERE DOB LIKE '1959%';

How many patients were born in 1959?
Answer

SELECT * FROM Patients
WHERE DOB LIKE '1959%';

12
1. What query would give you the number of patients born in August (of any year)? (Hint: study the data...)

2. How many patients were born in August (of any year)?
SELECT * FROM Patients
WHERE DOB LIKE '%-08-%';

How many patients were born in August (of any year)?
Answer

```
SELECT * FROM Patients
WHERE DOB LIKE '%%-08-%';
```

71
SELECT DISTINCT:
Selecting Each Variation
Only Once
SELECT DISTINCT

Consider this table – there are 6 patients with 3 different last names – Doe, Jones, and Smith

<table>
<thead>
<tr>
<th>ID</th>
<th>LastName</th>
<th>FirstName</th>
<th>StreetAddress</th>
<th>City</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Doe</td>
<td>John</td>
<td>123 Main Street</td>
<td>Houston</td>
</tr>
<tr>
<td>2</td>
<td>Jones</td>
<td>Mary</td>
<td>425 Allison Drive</td>
<td>Austin</td>
</tr>
<tr>
<td>3</td>
<td>Smith</td>
<td>Erica</td>
<td>327 Maple Drive</td>
<td>Austin</td>
</tr>
<tr>
<td>4</td>
<td>Jones</td>
<td>Marcus</td>
<td>54 Congress Ave.</td>
<td>Austin</td>
</tr>
<tr>
<td>5</td>
<td>Smith</td>
<td>Alan</td>
<td>26 Bluebell Court</td>
<td>Austin</td>
</tr>
<tr>
<td>6</td>
<td>Smith</td>
<td>Donald</td>
<td>874 Elizabeth Ave.</td>
<td>Austin</td>
</tr>
</tbody>
</table>
SELECT DISTINCT

What if you wanted to know how many different last names were in your Patients table? This won’t work because you don’t want all the names, just one entry for each name.

SELECT Last_Name FROM Patients;
SELECT DISTINCT

• SELECT DISTINCT returns only one row for each name, no matter how many times that name is present
  SELECT DISTINCT LastName
  FROM Patients;

• Your output will only have three lines:
  Doe
  Jones
  Smith
1. What query would give you the number of unique states in the Patients table?

2. How many states are in the Patients table?
SELECT DISTINCT State
FROM Patients;

How many states are in the Patients table?
SELECT DISTINCT State
FROM Patients;

49
Using BETWEEN:
Finding records in a range of values
Using BETWEEN

• What if you want to select only patients born between a specific range of dates?

• Syntax

```
SELECT column1, column2,...,columnN
FROM tablename
WHERE columnname BETWEEN number1 AND number2;
```
1. What query would give you a list of the last name, first name, gender, and weight of all patients who weigh between 100 and 150 pounds, including weights 100.00 and 150.00.

2. How many patients are on the list?
Answer

SELECT LastName, FirstName, Gender, Pounds
FROM Patients
WHERE Pounds BETWEEN 100 and 150;

2. How many patients are on the list?
Answer

SELECT LastName, FirstName, Gender, Pounds
FROM Patients
WHERE Pounds BETWEEN 100 and 150;

186
Using
> (greater than)
< (less than)
= (equal to)
Using >, <, -

• What if you want to select only patients who have a numeric value above or below a given number? (weight, Hgb A1C, blood pressure, etc.)

• Syntax

```sql
SELECT column1, column2,...columnN
FROM tablename
WHERE columnname operator value;
```

Operators: >, <, =, <=, >=
You Try It!

1. What query would give you a list of the last name, first name, gender, and weight of all patients who weigh 300 pounds or more?

2. How many patients are on the list?
SELECT * FROM Patients
WHERE Pounds >= 300;

2. How many patients are on the list?
Answer

SELECT * FROM Patients WHERE Pounds >= 300;

15
Using IN: Finding records from a specific set of values
Using IN

• What if you want to select only patients in a specific set of values?

• Syntax

```
SELECT column1, column2,...,columnN
FROM tablename
WHERE columnname IN (value1, value2,...,valueN);
```

Notice the parentheses
1. What query would give you a list of patients whose last name is one of the following: Jones, Thompson, Martinez, or Green.

2. How many patients are on the list?
Answer

SELECT * FROM Patients
WHERE LastName IN
('Jones', 'Thompson', 'Martinez', 'Green');

2. How many patients are on the list?
SELECT * FROM Patients
WHERE LastName IN
( 'Jones', 'Thompson', 'Martinez', 'Green' );

21
Sorting Records:
ORDER BY
ORDER BY

• ORDER BY is used to sort the results

• Syntax:

```sql
SELECT column_name(s)
FROM table_name
ORDER BY column_name(s) ASC [or DESC]
```

• Records can be sorted in ascending or descending order

  `ASC` is for ascending order
  `DESC` is for descending order
Sorting Records: ORDER BY

SELECT *
FROM Patients
WHERE LastName LIKE 'Smith'
ORDER BY DOB;
Aggregate Functions

Totaling, averaging, counting, identifying highest and lowest values
Aggregate Functions

• How many? COUNT()
• What is the average? AVG()
• What is the total? SUM()
• What is the highest value? MAX()
• What is the lowest value? MIN()

• () specifies either * or the specific column you are interested in
Using Aggregate Functions

• Syntax

```
SELECT functionname(columnname)
FROM tablename
WHERE CONDITION;
```

The WHERE is optional

Notice the parentheses
Counting Records
Using COUNT

• What if you want to just count the number of records and not get a list?

• Syntax

```
SELECT COUNT(columnname)  
FROM tablename  
WHERE CONDITION;
```

Notice the parentheses
1. What query would give you the number of patients in the Patients table?

2. What number did you get?
SELECT COUNT(*)
FROM Patients;

2. What number did you get?
SELECT COUNT(*)
FROM Patients;

790
A little more difficulty...

• Let’s try adding a WHERE clause!
1. What query would give you the number of male patients in the Patients table?

2. What number did you get?
Answer

SELECT COUNT(Gender)
FROM Patients
WHERE Gender LIKE 'M';

2. What number did you get?
Answer

SELECT COUNT(Gender) FROM Patients
WHEREGender LIKE 'M';

390
Calculating an Average
Averaging

• To average the values for a **specific** column

• Syntax

  SELECT AVG(columnname)
  FROM tablename
  WHERE CONDITION;

Notice the parentheses
1. What query would give you the average weight of all patients in the Patients table who are from Texas?
2. What number did you get?
Answer

```
SELECT AVG(Pounds) 
FROM Patients 
WHERE State LIKE 'TX';
```

2. What number did you get?
SELECT AVG(Pounds) FROM Patients WHERE State LIKE 'TX';

191
Let’s try another one
You Try It!

1. What query would give you the average weight of all patients in the Patients table who are female?

2. What number did you get?
Answer

```
SELECT AVG(Pounds) 
FROM Patients 
WHERE Gender LIKE 'F';
```

2. What number did you get?
SELECT AVG(Pounds) FROM Patients WHERE Gender LIKE 'F';

171.596473
Identifying the Minimum Value
Using MIN

• To identify the smallest value in a column:

• Syntax

```sql
SELECT MIN(columnname) FROM tablenam WHERE CONDITION;
```
You Try It!

1. What query would give you the lowest weight of all patients in the Patient table who are from Texas?

2. What number did you get?
Answer

```
SELECT MIN(Pounds)
FROM Patients
WHERE State LIKE 'TX';
```

2. What number did you get?
Answer

SELECT MIN(Pounds) FROM Patients WHERE State LIKE 'TX';

110.40
A little more difficulty...

• Let’s try adding more details!
1. What query would give you the last name, first name, and MRNO of the patient with the lowest weight?

2. What patient information did you get?
SELECT * 
FROM Patients 
WHERE Pounds LIKE '110.40';

2. What patient information did you get?
SELECT * 
FROM Patients 
WHERE Pounds LIKE '110.40';
Martha Swartz and Pamela Smyth
Identifying the Maximum Value
Using MAX

• To identify the largest value in a column:
• Syntax

```sql
SELECT MAX(columnname)
FROM tablename
WHERE CONDITION;
```

Notice the parentheses
You Try It!

1. What query would give you the highest weight of all patients in the Patients table who are from Texas?

2. What number did you get?
SELECT MAX(Pounds) FROM Patients WHERE State LIKE 'TX';

2. What number did you get?
SELECT MAX(Pounds) FROM Patients WHERE State LIKE 'TX';
451.50
CAUTION!

• Run this query:

SELECT *  
FROM Patients  
WHERE MRNO LIKE '6';

• How much does this patient weigh?
• Now run this query:

```sql
SELECT MAX(Pounds), MRNO
FROM Patients;
```

• Who did you get?
• What happened?
Aggregate Function Cautions

SUM, AVG, MIN, MAX, COUNT can do one task, not multiple

Cannot give a list as well as a result

WHERE does not give correct results when combined with aggregate functions
WHERE COUNT(BloodType) > 5

That’s a job for GROUP BY and HAVING!
Subtotals and Sorting:
GROUP BY and ORDER BY
Subtotals

- Let’s say you wanted to know how many patients had each kind of blood type:
  - A+, A-
  - B+, B-
  - AB+, AB-
  - O+, O-
Subtotals

You could do this, and write down the total number of records from each query:

```sql
SELECT * FROM Patients WHERE BloodType LIKE 'A+';
SELECT * FROM Patients WHERE BloodType LIKE 'A-';
SELECT * FROM Patients WHERE BloodType LIKE 'B+';
SELECT * FROM Patients WHERE BloodType LIKE 'B-';
SELECT * FROM Patients WHERE BloodType LIKE 'AB+';
SELECT * FROM Patients WHERE BloodType LIKE 'AB-';
SELECT * FROM Patients WHERE BloodType LIKE 'O+';
SELECT * FROM Patients WHERE BloodType LIKE 'O-';
```
Output from Multiple Queries

- But that’s a hassle!
So, you could add COUNT to your queries...

SELECT BloodType, COUNT(BloodType) FROM Patients WHERE BloodType LIKE 'A+';

<table>
<thead>
<tr>
<th>#</th>
<th>BloodType</th>
<th>COUNT(BloodType)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A+</td>
<td>190</td>
</tr>
</tbody>
</table>
But...

- You’d have to do this for every single blood type
- TIME and HASSLE!
- And what if you don’t know what all the values are in a given column?
GROUP BY

- GROUP BY is used with COUNT, MAX, MIN, SUM, and AVG to group results
- SELECT columnname(s)
  FROM tablename
  WHERE condition  \( \leftarrow \text{optional} \)
  GROUP BY columnname(s)
  ORDER BY columnname(s);  \( \leftarrow \text{optional} \)
1. What query would give you the blood types and the number of patients with each blood type?

2. What values did you get?
Answer

```
SELECT BloodType, COUNT(BloodType) 
FROM Patients 
GROUP BY BloodType;
```

2. What values did you get?
Answer

SELECT BloodType, COUNT(BloodType)
FROM Patients
GROUP BY BloodType;

<table>
<thead>
<tr>
<th>#</th>
<th>BloodType</th>
<th>COUNT(BloodType)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A+</td>
<td>190</td>
</tr>
<tr>
<td>2</td>
<td>A-</td>
<td>23</td>
</tr>
<tr>
<td>3</td>
<td>AB+</td>
<td>40</td>
</tr>
<tr>
<td>4</td>
<td>AB-</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>B+</td>
<td>193</td>
</tr>
<tr>
<td>6</td>
<td>B-</td>
<td>10</td>
</tr>
<tr>
<td>7</td>
<td>O+</td>
<td>297</td>
</tr>
<tr>
<td>8</td>
<td>O-</td>
<td>35</td>
</tr>
</tbody>
</table>
Now let’s sort this

How will you modify this so that the numbers are sorted from largest to smallest?

SELECT BloodType, COUNT(BloodType) FROM Patients GROUP BY BloodType;
SELECT BloodType, COUNT(BloodType)
FROM Patients
GROUP BY BloodType
ORDER BY COUNT(BloodType);

2. What output did you get?
ORDER BY defaults to ascending order

<table>
<thead>
<tr>
<th>#</th>
<th>BloodType</th>
<th>COUNT(BloodType)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>AB-</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>B-</td>
<td>10</td>
</tr>
<tr>
<td>3</td>
<td>A-</td>
<td>23</td>
</tr>
<tr>
<td>4</td>
<td>O-</td>
<td>35</td>
</tr>
<tr>
<td>5</td>
<td>AB+</td>
<td>40</td>
</tr>
<tr>
<td>6</td>
<td>A+</td>
<td>190</td>
</tr>
<tr>
<td>7</td>
<td>B+</td>
<td>193</td>
</tr>
<tr>
<td>8</td>
<td>O+</td>
<td>297</td>
</tr>
</tbody>
</table>
Add a DESC option to your ORDER BY:

```
SELECT BloodType, COUNT(BloodType)
FROM Patients
GROUP BY BloodType
ORDER BY COUNT(BloodType) DESC;
```

2. Now what output did you get?
Addition of DESC to ORDER BY clause

<table>
<thead>
<tr>
<th>#</th>
<th>BloodType</th>
<th>COUNT(BloodType)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>O+</td>
<td>297</td>
</tr>
<tr>
<td>2</td>
<td>B+</td>
<td>193</td>
</tr>
<tr>
<td>3</td>
<td>A+</td>
<td>190</td>
</tr>
<tr>
<td>4</td>
<td>AB+</td>
<td>40</td>
</tr>
<tr>
<td>5</td>
<td>O-</td>
<td>35</td>
</tr>
<tr>
<td>6</td>
<td>A-</td>
<td>23</td>
</tr>
<tr>
<td>7</td>
<td>B-</td>
<td>10</td>
</tr>
<tr>
<td>8</td>
<td>AB-</td>
<td>2</td>
</tr>
</tbody>
</table>
Filtering: HAVING
Why HAVING is important

Remember that

SELECT MAX(Pounds), MRNO
FROM Patients;

Gave you the wrong result?
HAVING

• Must use GROUP BY
• Can use any of the aggregate functions to filter: SUM, AVG, MAX, MIN, COUNT
HAVING Syntax

SELECT columnname(s)
FROM tablename
WHERE condition \(\leftarrow\) optional
GROUP BY columnname(s) \(\leftarrow\) mandatory
for HAVING to work
HAVING condition
ORDER BY columnname(s); \(\leftarrow\) optional
1. Run this:

```sql
SELECT BloodType, COUNT(BloodType) 
FROM Patients 
GROUP BY BloodType 
HAVING COUNT(BloodType) > 100;
```

2. What output did you get?
1. Run this:

```sql
SELECT BloodType, COUNT(BloodType) 
FROM Patients 
GROUP BY BloodType 
HAVING COUNT(BloodType) > 100;
```

<table>
<thead>
<tr>
<th>#</th>
<th>BloodType</th>
<th>COUNT(BloodType)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A+</td>
<td>190</td>
</tr>
<tr>
<td>2</td>
<td>B+</td>
<td>193</td>
</tr>
<tr>
<td>3</td>
<td>O+</td>
<td>297</td>
</tr>
</tbody>
</table>
Logical Operators (Boolean Logic)
Boolean Logic

• These were examples of Boolean logic in use by a computer system
• Named after 19th century mathematician, George Boole
• Fundamental to computer operation and data retrieval
Refresher: Blood Types

• Humans have four blood groups: A, B, AB, and O
• They may also have an Rh factor, and presence or absence is indicated with a + or –
• So blood types are A+, A-, B+, B-, AB+, AB-, O+, and O-.
Refresher: Simple SELECT

- You know how to retrieve records that match a particular criterion or condition

```sql
SELECT [ what – columns or * for all columns ]
FROM [ location- name of table ]
WHERE [ condition ] ;
```
But what if you need combinations?

• What if you need multiple conditions, say, Patients >= 21 years old AND who are blood type A+?

• You know how to do this:
  – SELECT * FROM Patients WHERE BloodType like ‘A+’;
  – SELECT * FROM Patients WHERE AGE >= 21;

• But you wind up with 2 separate lists and you need one combined list with only those patients who fit both criteria

• How do you describe the rules for what you want?
Retrieving Records Using Logical Arguments (Boolean logic)

• Patients who are >= 21 years old and who are blood type A+?
Retrieving Records Using Logical Arguments (Boolean logic)

- Patients who are \( \geq 21 \) years old and who are blood type A+?

```
WHERE BloodType LIKE 'A+' AND Age >= 21
```
Boolean Operator: AND
Simple Example

• A research study is recruiting patients to test a new treatment. Subjects must be \geq 21\text{ years old AND have a blood type of A+}.

• You have to produce a list of all patients in the database who meet these criteria.
Simple Example

A research study is recruiting patients to test a new treatment. Subjects must be $\geq 21$ years old AND have a blood type of A+.

```
SELECT * FROM Patients
WHERE AGE $\geq 21$ AND
BloodType LIKE 'A+';
```
The Logic
The Logic

• Each row (record) in the database is evaluated to see if it meets the condition(s) specified in the WHERE clause.

• A record will ONLY appear in your list if it passes that test (meets the conditions).

• You need to understand the data in your database! In real life, queries must be tested before using them for production.
Simple Example

A research study is recruiting patients to test a new treatment. Subjects must be $\geq 21$ years old AND have a blood type of A+.

<table>
<thead>
<tr>
<th>$\geq 21?$</th>
<th>Has blood type A+?</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Simple Example

A research study is recruiting patients to test a new treatment. Subjects must be $\geq 21$ years old AND have a blood type of A+.

<table>
<thead>
<tr>
<th>$\geq 21?$</th>
<th>Has blood type A+?</th>
</tr>
</thead>
<tbody>
<tr>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>
A research study is recruiting patients to test a new treatment. Subjects must be $\geq 21$ years old AND have a blood type of A+.

<table>
<thead>
<tr>
<th>$\geq 21?$</th>
<th>Has blood type A+?</th>
</tr>
</thead>
<tbody>
<tr>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Simple Example

A research study is recruiting patients to test a new treatment. Subjects must be $\geq 21$ years old AND have a blood type of A+.

<table>
<thead>
<tr>
<th>$\geq 21?$</th>
<th>Has blood type A+?</th>
<th>Can participate?</th>
</tr>
</thead>
<tbody>
<tr>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Simple Example

A research study is recruiting patients to test a new treatment. Subjects must be >= 21 years old AND have a blood type of A+.

<table>
<thead>
<tr>
<th>&gt;= 21?</th>
<th>Has blood type A+?</th>
<th>Can participate?</th>
</tr>
</thead>
<tbody>
<tr>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Simple Example

A research study is recruiting patients to test a new treatment. Subjects must be $\geq 21$ years old AND have a blood type of A+.

<table>
<thead>
<tr>
<th>$\geq 21?$</th>
<th>Has blood type A+?</th>
<th>Can participate?</th>
</tr>
</thead>
<tbody>
<tr>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
</tbody>
</table>
Simple Example

A research study is recruiting patients to test a new treatment. Subjects must be $\geq 21$ years old AND have a blood type of A+.

<table>
<thead>
<tr>
<th>$\geq 21?$</th>
<th>Has blood type A+?</th>
<th>Can participate?</th>
</tr>
</thead>
<tbody>
<tr>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>
A research study is recruiting patients to test a new treatment. Subjects must be \( \geq 21 \) years old AND have a blood type of A+. Both conditions must be satisfied, or the record won’t appear in our list of eligible patients.

<table>
<thead>
<tr>
<th>( \geq 21? )</th>
<th>Has blood type A+?</th>
<th>Can participate?</th>
</tr>
</thead>
<tbody>
<tr>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Truth Tables

• Mathematical table used to tell whether an expression is true for all legitimate input values

• Shows the expected results for a given set of input conditions and an operator (such as AND or OR)
Why is this important?

• Writing out the logic will help you when you get ready to write your SQL statement.

• Also, you need to understand what results you expect to get
1. You are working in a large family practice and one of the physicians has realized that a tall, slender man she saw recently may have Marfan syndrome. She can't remember the patient's name.

What query would produce a list of all male patients who weigh less than 150 pounds, sorted so the tallest patient is listed first?

2. What is the medical record number (MRNO) of the tallest patient?
SELECT * FROM Patients
WHERE Gender LIKE 'M' AND
Pounds < 150
ORDER BY FeetInches DESC;

2. What is the medical record number (MRNO) of the tallest patient?
SELECT * FROM Patients
WHERE Gender LIKE 'M' AND
Pounds < 150
ORDER BY FeetInches DESC;

5154
A Little Bit More Difficult
1. You are working in the blood bank in a local hospital that handles a high number of obstetrics patients.

What query would produce a list of patients who are female and have an Rh negative blood type (such as A-), sorted by MRNO?

2. How many patients are on the list?
Answer

SELECT * FROM Patients
WHERE Gender LIKE 'F'
AND BloodType LIKE '%-'
ORDER BY MRNO;

2. How many patients are on the list?
Answer

```sql
SELECT * FROM Patients
WHERE Gender LIKE 'F'
AND BloodType LIKE '\%-'
ORDER BY MRNO;
```

42
Boolean Operator: NOT
The Long Way to Exclude Something, 1

• To select all data from a table named PATIENTS for patients who have any blood type except A+

• You could build a set of the blood types you want using IN — but what are the drawbacks?

```
SELECT MRNO, LastName, FirstName, Gender, BloodType
FROM Patients
WHERE BloodType IN ('A-', 'B+', 'B-', 'AB+', 'AB-', 'O+', 'O-');
```
The Long Way to Exclude Something, 2

Why might this not be the best way?

Where BloodType IN
('A-', 'B+', 'B-', 'AB+', 'AB-', 'O+', 'O-');

Think of the possibility of:

-- misspellings
-- leaving out something**
-- accidentally including something
Using NOT in SQL

- A faster way to select all data from a table named PATIENTS for patients who have any blood type except A+: specify exactly what you DON’T want:

```
SELECT * FROM Patients
WHERE BloodType NOT LIKE 'A+';
```
1. Your director comes to you with a problem. Apparently, the number of male patients and female patients does not add up to the total number of patients in the system. What query would produce a list of all patients who have a gender other than M or F?

2. How many patients are on the list?
Answer

SELECT *
FROM Patients
WHERE Gender NOT IN ('M', 'F');

2. How many patients are on the list?
SELECT * 
FROM Patients 
WHERE Gender NOT IN ('M', 'F');
3
Important!

More than one approach may work

*You have to understand your data and evaluate what syntax will be best for the question you are trying to answer given the circumstances you are working under*
Boolean Operator: OR
OR: Example

• The research study now decides it will take patients if they have a blood type of A+ or AB+.
OR Operator in SQL

• To select all data from a table named Patients who have the last name of Smith OR have the last name of Jones:

```sql
SELECT * FROM Patients
WHERE LastName LIKE 'Smith' OR
   LastName LIKE 'Jones';
```
Run this query:

```
SELECT * FROM Patients
WHERE Employer LIKE '%lawn%' 
OR Employer LIKE '%garden%' 
OR Employer LIKE '%yard%';
```

How many records did you get? 191
Now run this query:

```
SELECT * FROM Patients
WHERE Employer IN ('%lawn%', '%garden%', '%yard%');
```

How many records did you get?
CAUTION!

SQL cannot do two things at the same time – it cannot “fill in” wildcards in a set, and also compare a value against that.

If you would like a detailed explanation, see http://stackoverflow.com/questions/1127088/mysql-like-in.

Understand your data and use another method to cross-check your results, especially while you are learning.
Putting Operators Together
Putting Operators Together

- Operators can be put together to build expressions
- For example: \((\text{NOT } X) \text{ OR } (\text{NOT } Y)\)
- This reads NOT X OR NOT Y

<table>
<thead>
<tr>
<th>Condition X</th>
<th>Condition Y</th>
<th>NOT X</th>
<th>NOT Y</th>
<th>(NOT X) OR (NOT Y)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Example

SELECT * FROM Patients
WHERE
(BloodType NOT LIKE 'B+') AND [or OR]
(Gender NOT LIKE 'M');

Just as in mathematics, there is an order or sequence of how the equation is evaluated. What is inside the parentheses is first, and then the operator outside the parentheses.
Organizing Your Work

SELECT LastName, FirstName, Gender, State FROM Patients
WHERE Gender LIKE 'M' AND State LIKE 'AZ' OR Gender LIKE 'F' AND State LIKE 'KY'
ORDER BY LastName;

Not very good!
SELECT LastName, FirstName, Gender, State
FROM Patients
WHERE
   (State LIKE 'AZ' AND Gender LIKE 'M')
OR
   (State LIKE 'KY' AND Gender LIKE 'F')
ORDER BY LastName;

• Explicitly states the sequence of how things should be evaluated
• Less chance for error
• Easier to read and for others to understand

Parentheses are your friends!!
Identifiers and Keys
Reducing Redundancy

- 2 records for John Doe (1 for each of his doctors)
- What if we update John Doe’s address?
- What if we want to delete John Doe?

<table>
<thead>
<tr>
<th>ID</th>
<th>Last_name</th>
<th>First_name</th>
<th>Street_Address</th>
<th>City</th>
<th>Dr_lname</th>
<th>Dr_fname</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Doe</td>
<td>John</td>
<td>123 Main Street Houston</td>
<td>Houston</td>
<td>Jones</td>
<td>John</td>
</tr>
<tr>
<td>2</td>
<td>Jones</td>
<td>Mary</td>
<td>425 Allison Dr Austin</td>
<td>Austin</td>
<td>Garcia</td>
<td>Lupe</td>
</tr>
<tr>
<td>3</td>
<td>Smith</td>
<td>Erica</td>
<td>327 Maple Drive Austin</td>
<td>Austin</td>
<td>Gordon</td>
<td>Bill</td>
</tr>
<tr>
<td>4</td>
<td>Doe</td>
<td>John</td>
<td>123 Main Street Houston</td>
<td>Houston</td>
<td>Gordon</td>
<td>Bill</td>
</tr>
</tbody>
</table>
How Database Designers Fix This, 1

• Break the table up into 2 tables – *each with just 1 theme*
  – Patient table
  – Doctor table

<table>
<thead>
<tr>
<th>ID</th>
<th>Last_name</th>
<th>First_name</th>
<th>Street_Address</th>
<th>City</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Doe</td>
<td>John</td>
<td>123 Main Street</td>
<td>Houston</td>
</tr>
<tr>
<td>2</td>
<td>Jones</td>
<td>Mary</td>
<td>425 Allison Drive</td>
<td>Austin</td>
</tr>
<tr>
<td>3</td>
<td>Smith</td>
<td>Erica</td>
<td>327 Maple Drive</td>
<td>Austin</td>
</tr>
<tr>
<td>4</td>
<td>Doe</td>
<td>John</td>
<td>123 Main Street</td>
<td>Houston</td>
</tr>
</tbody>
</table>
How Database Designers Fix This, 2

• Then, link the two tables together by means of an identifier

<table>
<thead>
<tr>
<th>ID</th>
<th>Dr_Iname</th>
<th>Dr_fname</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Jones</td>
<td>John</td>
</tr>
<tr>
<td>2</td>
<td>Garcia</td>
<td>Lupe</td>
</tr>
<tr>
<td>3</td>
<td>Gordon</td>
<td>Bill</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ID</th>
<th>Last_name</th>
<th>First_name</th>
<th>Street_Address</th>
<th>City</th>
<th>PrimaryMD</th>
<th>SecondaryMD</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Doe</td>
<td>John</td>
<td>123 Main Street</td>
<td>Houston</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>Jones</td>
<td>Mary</td>
<td>425 Allison Drive</td>
<td>Austin</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Smith</td>
<td>Erica</td>
<td>327 Maple Drive</td>
<td>Austin</td>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>
Identifiers

• Think about how the following things are identified:
  – Cars
  – Patients
  – Computers

• Databases have to have a way to uniquely identify a single record in a table
Keys

• Every entry in a database must have a unique identifier, or key

• A key is an attribute or a group of attributes, that has a unique value for each entry in a table.

<table>
<thead>
<tr>
<th>PATIENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Med_rec_number</td>
</tr>
<tr>
<td>First_name</td>
</tr>
<tr>
<td>Last_name</td>
</tr>
<tr>
<td>Date_of_birth</td>
</tr>
<tr>
<td>Street_address</td>
</tr>
<tr>
<td>City</td>
</tr>
<tr>
<td>State</td>
</tr>
<tr>
<td>Zip</td>
</tr>
<tr>
<td>Driver_license_number</td>
</tr>
<tr>
<td>Social_security_number</td>
</tr>
<tr>
<td>Phone_number_home</td>
</tr>
<tr>
<td>Phone_number_work</td>
</tr>
</tbody>
</table>
Types of Keys

• **Primary key**: a unique identifier chosen as the key used to uniquely identify a row.

• **Foreign key**: a primary key of one table that is placed in a second table.
### Primary and Foreign Keys

Each table has an ID column. The ID from the Doctor table is inserted as a Foreign Key in the Primary Doctor column in the Patient table.

<table>
<thead>
<tr>
<th>ID</th>
<th>Last_name</th>
<th>First_name</th>
<th>Street_Address</th>
<th>City</th>
<th>Dr_Primary</th>
<th>Dr_Secondary</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Doe</td>
<td>John</td>
<td>123 Main Street</td>
<td>Houston</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>Jones</td>
<td>Mary</td>
<td>425 Allison Drive</td>
<td>Austin</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Smith</td>
<td>Erica</td>
<td>327 Maple Drive</td>
<td>Austin</td>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ID</th>
<th>Dr_First_name</th>
<th>Dr_Last_name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>John</td>
<td>Jones</td>
</tr>
<tr>
<td>2</td>
<td>Lupe</td>
<td>Garcia</td>
</tr>
<tr>
<td>3</td>
<td>Bill</td>
<td>Gordon</td>
</tr>
</tbody>
</table>
Why this important

• This insertion of an identifier from one table into another as a foreign key creates a chain, allowing us to retrieve data across tables
You Try It!

1. Run a query for all records from the Diagnoses table

2. What can you tell about the first patient on the list?
SELECT * 
FROM Diagnoses; 

2. What can you tell about the first patient on the list?
SELECT * 
FROM Diagnoses;

ID number 1, medical record number 343, and an ICD 10 code and description
You Try It!

1. Now run a query for this patient’s record from the Patients table

2. Do you see any diagnosis data?
SELECT * 
FROM Patients 
WHERE MRNO LIKE '343';

2. Do you see any diagnosis data?
Answer

SELECT * 
FROM Patients 
WHERE MRNO LIKE '343';

No
Retrieving Data from Multiple Tables: Using JOIN
JOIN

• Join statements use a data element that is present in both tables to “link” the tables together

• This allows the query to produce the data the user wants, even though the data are spread across multiple tables
JOIN: Example

**Patients Table**

<table>
<thead>
<tr>
<th>MRNO</th>
<th>LastName</th>
<th>FirstName</th>
<th>MiddleInit</th>
<th>DOB</th>
</tr>
</thead>
<tbody>
<tr>
<td>343</td>
<td>Curran</td>
<td>Barbara</td>
<td>N</td>
<td>1962-04-19</td>
</tr>
</tbody>
</table>

Is using these two fields to link the tables together

**Diagnoses Table**

<table>
<thead>
<tr>
<th>ID</th>
<th>MRNO</th>
<th>DX1</th>
<th>Description 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>343</td>
<td>W59.21XA</td>
<td>Bitten by turtle, initial encounter</td>
</tr>
</tbody>
</table>
JOIN: Syntax

- We have to tell SQL what to match on
- And since “MRNO” can be in multiple places, we also have to explicitly give the name of the table that we want it to look at
- (Just like “John Smith” vs. just “John”)
JOIN: Syntax

SELECT * 
FROM Patients 
WHERE MRNO LIKE ‘343’;

JOIN: Syntax

SELECT *  
FROM Patients JOIN Diagnoses  
WHERE MRNO LIKE ‘343’;
JOIN: Syntax

SELECT *  
FROM Patients JOIN Diagnoses 
on Patients.MRNO=Diagnoses.MRNO 
WHERE MRNO LIKE ‘343’;
JOIN: Syntax

```
SELECT * 
FROM Patients JOIN Diagnoses 
on Patients.MRNO=Diagnoses.MRNO 
WHERE Patients.MRNO LIKE ‘343’;
```
JOIN (also called INNER JOIN)

Gives only the set of records that match in both the Patients table and the Diagnoses table.
JOIN: Result

• Only patients that also have an entry in the Diagnoses table will appear
You Try It!

```
SELECT * 
FROM Patients
WHERE MRNO LIKE '343';
```

Scroll to the right and look at the columns – do you see any diagnosis data?
You Try It!

SELECT * 
FROM Patients JOIN Diagnoses  
on Patients.MRNO=Diagnoses.MRNO  
WHERE Patients.MRNO LIKE ‘343’;
Answer

Scroll through your output and look at the output
You Try It!

1. Retrieve all the data from the Patients table and the Diagnoses table for only those patients who have the code **G40.901** in the **DX1** column. (This is the code for "Epilepsy unspecified").

2. How many patients did you get?
SELECT * FROM Patients
JOIN Diagnoses ON Patients.MRNO=Diagnoses.MRNO
WHERE DX1 LIKE 'G40.901';

2. How many patients did you get?
SELECT * FROM Patients JOIN Diagnoses ON Patients.MRNO=Diagnoses.MRNO WHERE DX1 LIKE 'G40.901';

8
1. Retrieve all the data from the Patients table and the Diagnoses table for only those patients who have an ICD10 code E11.9 (the code for diabetes mellitus) for DX1, DX2, or DX3.

2. How many patients did you get?
SELECT * FROM Patients
JOIN Diagnoses ON Patients.MRNO=Diagnoses.MRNO
WHERE
(DX1 LIKE 'E11.9' OR DX2 LIKE 'E11.9' OR DX3 LIKE '11.9');

2. How many patients did you get?
Answer

SELECT * FROM Patients
JOIN Diagnoses ON Patients.MRNO=Diagnoses.MRNO
WHERE
(DX1 LIKE 'E11.9' OR DX2 LIKE 'E11.9' OR DX3 LIKE 'E11.9');

20
LEFT JOIN
(sometimes also called Left Outer Join)
LEFT JOIN

• LEFT JOIN returns all records from the left table (table_name1), even if there are no matches in the right table (table_name2)

• If there are no matching records from table_name2, then those fields will be empty

(From http://www.w3schools.com/sql/sql_join_left.asp)
LEFT JOIN (Venn Representation)

• Gives all records from Table A, with any matching records from Table B.
• If there are no matching records from Table B, then those fields will be empty.
LEFT JOIN

• Syntax:

```sql
SELECT column_name(s)
FROM table_name1  LEFT JOIN table_name2
ON table_name1.column_name=table_name2.column_name
```

(From http://www.w3schools.com/sql/sql_join_left.asp)
You Try It!

1. SELECT *
   FROM Patients LEFT JOIN Diagnoses ON Patients.MRNO=Diagnoses.MRNO;

2. How many patients did you get?
1. SELECT * 
   FROM Patients LEFT JOIN Diagnoses ON Patients.MRNO=Diagnoses.MRNO;

790
# JOINS - Summary

<table>
<thead>
<tr>
<th>LEFT JOIN</th>
<th>RIGHT JOIN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gives all records from Table A, with any matching records from Table B. If there is no match, the right side will be empty.</td>
<td>Gives all records from Table B, with any matching records from Table A. If there is no match, the left side will be empty.</td>
</tr>
</tbody>
</table>
RIGHT JOIN
(sometimes also called Right Outer Join)
RIGHT JOIN

• RIGHT JOIN returns all records from the right table (table_name2), even if there are no matches in the left table (table_name1)
• If there are no matching records from table_name1, then those fields will be empty
• Syntax:

SELECT column_name(s)
FROM table_name1 RIGHT JOIN table_name2
ON table_name1.column_name=table_name2.column_name

(From http://www.w3schools.com/sql/sql_join_right.asp)
RIGHT JOIN

• Gives all records from Table B, with any matching records from Table A. If there is no match, the left side will be empty.
You Try It!

1. SELECT * FROM Patients RIGHT JOIN Diagnoses ON Patients.MRNO=Diagnoses.MRNO;

2. How many patients did you get?
1. SELECT * FROM Patients RIGHT JOIN Diagnoses ON Patients.MRNO=Diagnoses.MRNO;

800 – the Diagnoses table has 10 rows that don’t match to any patient!
Creating a Table
Requirements for Tables

• A name for the table
• What data elements (attributes) will be stored in the table?
• How will those be named?
• What kind of format will be required for each kind of data element?
Data Types

• How does the computer “know” what we are trying to store?
  – Street address: “123 Main Street”
  – Zip code: “78621”
  – Text: “There is no evidence of malignancy”
  – Lab result: Potassium = 4.2
  – Weight: 132 pounds
  – Image: X-ray
  – Boolean: Does the patient smoke?
Data Types

- We have to tell the computer what we are trying to store
- *Data type* = A set of values and a valid set of operations (such as addition, subtraction, multiplication) that are allowed on those values)

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>char</td>
<td>Character string</td>
<td>John Doe</td>
</tr>
<tr>
<td>Int</td>
<td>Integer</td>
<td>Whole number, can be positive or negative</td>
</tr>
<tr>
<td>bool</td>
<td>Boolean</td>
<td>True or False</td>
</tr>
<tr>
<td>float</td>
<td>Floating point number</td>
<td>Numbers with decimal points</td>
</tr>
</tbody>
</table>
How would a computer understand?

• The computer needs to be told what kind of data each field is, so that it stores and retrieves it correctly
• This is the concept of a data type
• Names, credit card numbers, yes/no answers – each has its own data type
• Software engineers and database developers specify the appropriate data type for each type of data that is being used.
Consider an Online Form

- Possible data types for the different fields

  - Char or string
  - Date
  - Boolean
  - Might be integer or character
Let’s create a Providers table

• Table name: Providers
• We want to store the following data elements:
  – ID number
  – First name
  – Last name
  – Specialty
  – Phone number
# Defining Data Elements

<table>
<thead>
<tr>
<th>Data Element</th>
<th>Data Type</th>
<th>How many characters?</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID number</td>
<td>Integer (whole numbers only)</td>
<td>20</td>
</tr>
<tr>
<td>First name</td>
<td>Letters and numbers (“varchar”)</td>
<td>50</td>
</tr>
<tr>
<td>Last name</td>
<td>Letters and numbers (“varchar”)</td>
<td>50</td>
</tr>
<tr>
<td>Specialty</td>
<td>Letters and numbers (“varchar”)</td>
<td>50</td>
</tr>
<tr>
<td>Phone</td>
<td>Letters and numbers (“varchar”)</td>
<td>15</td>
</tr>
</tbody>
</table>
What will be the primary key?

<table>
<thead>
<tr>
<th>Data Element</th>
<th>Data Type</th>
<th>How many characters?</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID number</td>
<td>Integer (whole numbers only)</td>
<td>20</td>
</tr>
<tr>
<td>First name</td>
<td>Letters and numbers (“varchar”)</td>
<td>50</td>
</tr>
<tr>
<td>Last name</td>
<td>Letters and numbers (“varchar”)</td>
<td>50</td>
</tr>
<tr>
<td>Specialty</td>
<td>Letters and numbers (“varchar”)</td>
<td>50</td>
</tr>
<tr>
<td>Phone</td>
<td>Letters and numbers (“varchar”)</td>
<td>15</td>
</tr>
</tbody>
</table>
CREATE TABLE Providers
(
DrIDNumber integer(20) PRIMARY KEY,
DrFirstName varchar(50),
DrLastName varchar(50),
DrSpecialty varchar(50),
DrPhone varchar(15)
);
Check to see if your table was created

DESCRIBE Providers;

• Also check the Schemas list – click Refresh if your table isn’t displaying
You Try It!

What is in your Providers table?

SELECT * FROM Providers;
We need to add some providers!

• Syntax:

```sql
INSERT INTO tablename
(column1, column2 .... columnN)
VALUES
(‘value1’, ‘value2’.... ‘valueN’);
```
INSERT INTO Providers 
(DrIDNumber, DrFirstName, DrLastName, DrSpecialty, DrPhone)
VALUES 
('75623', 'Loren', 'Morgan', 'Nephrology', '713-555-3465'),
('24852', 'Frank', 'Wagoner', 'Obstetrics', '713-555-3572');

You Try It!
You Try It!

• Now run a query to retrieve all the records from the Providers table
Answer

```
SELECT * FROM Providers;
```

<table>
<thead>
<tr>
<th>#</th>
<th>DrIDNumber</th>
<th>DrFirstName</th>
<th>DrLastName</th>
<th>DrSpecialty</th>
<th>DrPhone</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>24852</td>
<td>Frank</td>
<td>Wagoner</td>
<td>Obstetrics</td>
<td>713-555-3572</td>
</tr>
<tr>
<td>2</td>
<td>75623</td>
<td>Loren</td>
<td>Morgan</td>
<td>Nephrology</td>
<td>713-555-3465</td>
</tr>
<tr>
<td>*</td>
<td>NULL</td>
<td>NULL</td>
<td>NULL</td>
<td>NULL</td>
<td>NULL</td>
</tr>
</tbody>
</table>
Updating a Record

• Syntax:

UPDATE tablename
SET columnname = ’new value'
WHERE columnname LIKE ‘value’;
You Try It!

- You realize that you entered the specialty for Dr Morgan incorrectly; he is a surgeon, not a nephrologist. Change his specialty to SURGERY.
UPDATE Providers
SET DrSpecialty='SURGERY'
WHERE DrFirstName LIKE 'LOREN' AND DrLastName LIKE 'Morgan' AND DrIDNumber = '75623';
If we have time:
Date Calculations
Simple Calculations and Date Differences

• You’ll modify your Patients table to add columns to store some financial data as well as discharge date.

• Next, you’ll work with one of your fake patients, who was admitted a few years ago. You’ll do some calculations, such as her balance (how much she owes), her current age, and how many days she’s been in the hospital.
1. Add a Starting Balance column to the Patients table

ALTER TABLE Patients
ADD Start_balance decimal(10,2);
2. Add an End Balance column to the Patients table

```sql
ALTER TABLE Patients
ADD End_balance decimal(10,2);
```
3. Add a Discharge Date column to the Patients table

ALTER TABLE Patients
ADD DischargeDate date;
4. Check your work

Check the table structure to confirm that these columns are now present [scroll all the way down to see the new columns]

```
DESCRIBE Patients;
```
Set Edith’s start balance

UPDATE Patients
SET Start_balance='3472.28'
WHERE MRNO = '8821' AND FirstName LIKE 'Edith' AND LastName LIKE 'Luna';

Tip: Check that the Message area in the bottom right pane shows "1 row(s) affected"
Set Edith’s end balance

UPDATE Patients
SET End_balance='25342.96'
WHERE
MRNO = '8821' AND
FirstName LIKE 'Edith' AND
LastName LIKE 'Luna';

Tip: Check that the Message area in the bottom right pane shows "1 row(s) affected"
Check your work

Check Edith's record and scroll to the right to make sure the start and end balances have been updated:

```
SELECT * FROM Patients
WHERE
MRNO = '8821' AND
FirstName LIKE 'Edith' AND
LastName LIKE 'Luna';
```
How much does Edith owe?

```
SELECT *, (End_balance - Start_balance) AS Visit_Balance
FROM Patients
WHERE
MRNO = '8821' AND
FirstName LIKE 'Edith' AND
LastName LIKE 'Luna';
```
If we have time:
Date Calculations: NOW
NOW Function

We’ll use the NOW function to compute how many days Edith Luna has been in the hospital, from admission to today (you should be able to estimate whether this answer is right)
SELECT FirstName, LastName, AdmitDate, NOW() AS Today, DATEDIFF (NOW(), AdmitDate) AS DaysInHospital FROM Patients WHERE MRNO = '8821' AND FirstName LIKE 'Edith' AND LastName LIKE 'Luna';
Summary

• Used DESCRIBE, SELECT, and WHERE to perform simple queries
• Used pattern matching with LIKE, NOT LIKE, and =, and use Boolean operators AND, OR, and NOT
• Performed basic mathematical functions with SQL, such as averages and counts
• Used JOIN to combine data from two tables
• Must understand your data!
• All databases have errors; have a way to check your work
Questions?

List of additional training resources is on my web page

Thank you!
If we have time:
Stored Procedures
What is a stored procedure?

• Creating a stored procedure
The stored procedure definition screen

```
-- Routine DDL
-- Note: comments before and after the routine body will not be stored by the server
DELIMITER $$
CREATE PROCEDURE 'new_procedure' ()
BEGIN
END
```
CREATE PROCEDURE 'find_donor' (IN btype VARCHAR(255), IN sex VARCHAR(255))
BEGIN
SELECT * FROM Patients WHERE BloodType = btype AND Gender = sex;
END
Save your stored procedure

- Click Apply

```
DELIMITER $$

CREATE PROCEDURE `find_donor` (IN btype VARCHAR(255), IN sex VARCHAR(255))
BEGIN
  SELECT * FROM Patients WHERE BloodType = btype AND Gender = sex;
END$$
```
Confirm procedure is saved
Run the procedure

• `call find_donor('A-', 'F');`

• How many patients A- female patients did you get?

• Run the `find_donor` stored procedure again, but this time search for male patients who are B+. How many B+ male patients did you get?
Run the procedure

• call find_donor('A-','F');

• How many patients A- female patients did you get? 17

• Run the find_donor stored procedure again, but this time search for male patients who are B+. How many B+ male patients did you get? 102