

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

The screenshot shows the MySQL Workbench interface. On the left, the 'SCHEMAS' panel has 'Assignment' selected. The main editor window is titled 'SQL File 1 x' and contains a single line with the number '1'. A lightning bolt icon is visible in the toolbar above the editor. Below the editor is the 'Action Output' panel, which is currently empty. Five red callout boxes with white text provide instructions:

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

Added new script editor

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

Explanation

Syntax:

```
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

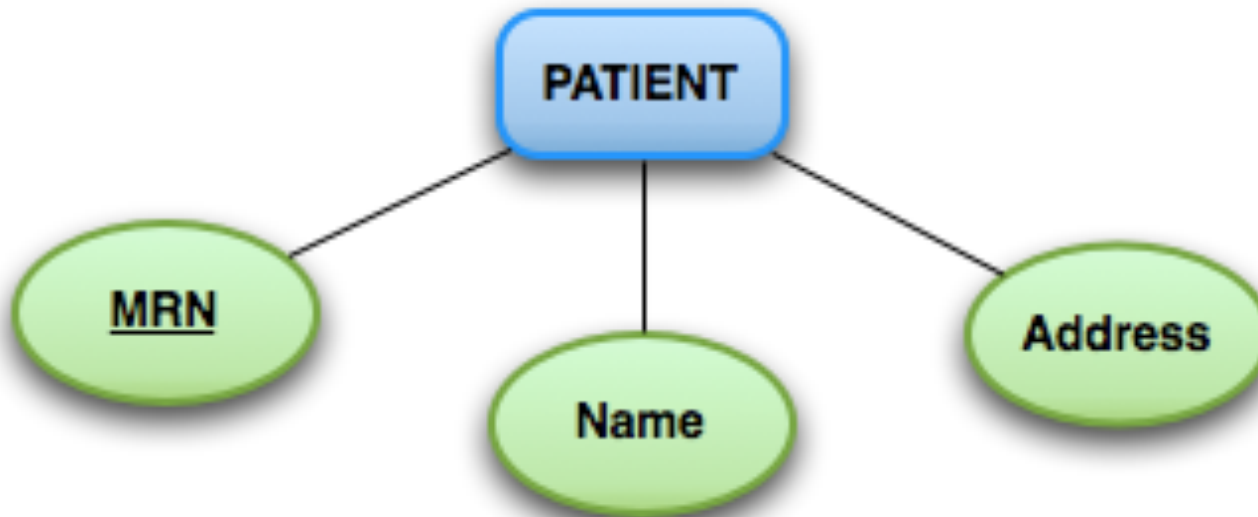
- Use **single quote marks** around groups of letters (these are called *strings*)
- **Do not use double quote marks**
- Copying and pasting from Word into MySQL can cause problems because Word uses a different character called a smart quote – best to just type in the quote marks directly in MySQL
- 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 ...”

Tables

Attributes or Properties



PATIENT

Medical record #

Name

Address

Table with Data

- Tables have two parts
 1. Heading (or columns)
 2. Body (or rows)

ID	Last_name	First_name	Street_Address	City
1	Doe	John	123 Main Street	Houston
2	Jones	Mary	425 Allison Drive	Austin
3	Smith	Erica	327 Maple Drive	Austin

Column headings

Rows

Table Attributes or Properties

The name of the table

The primary identifier (key) for this table

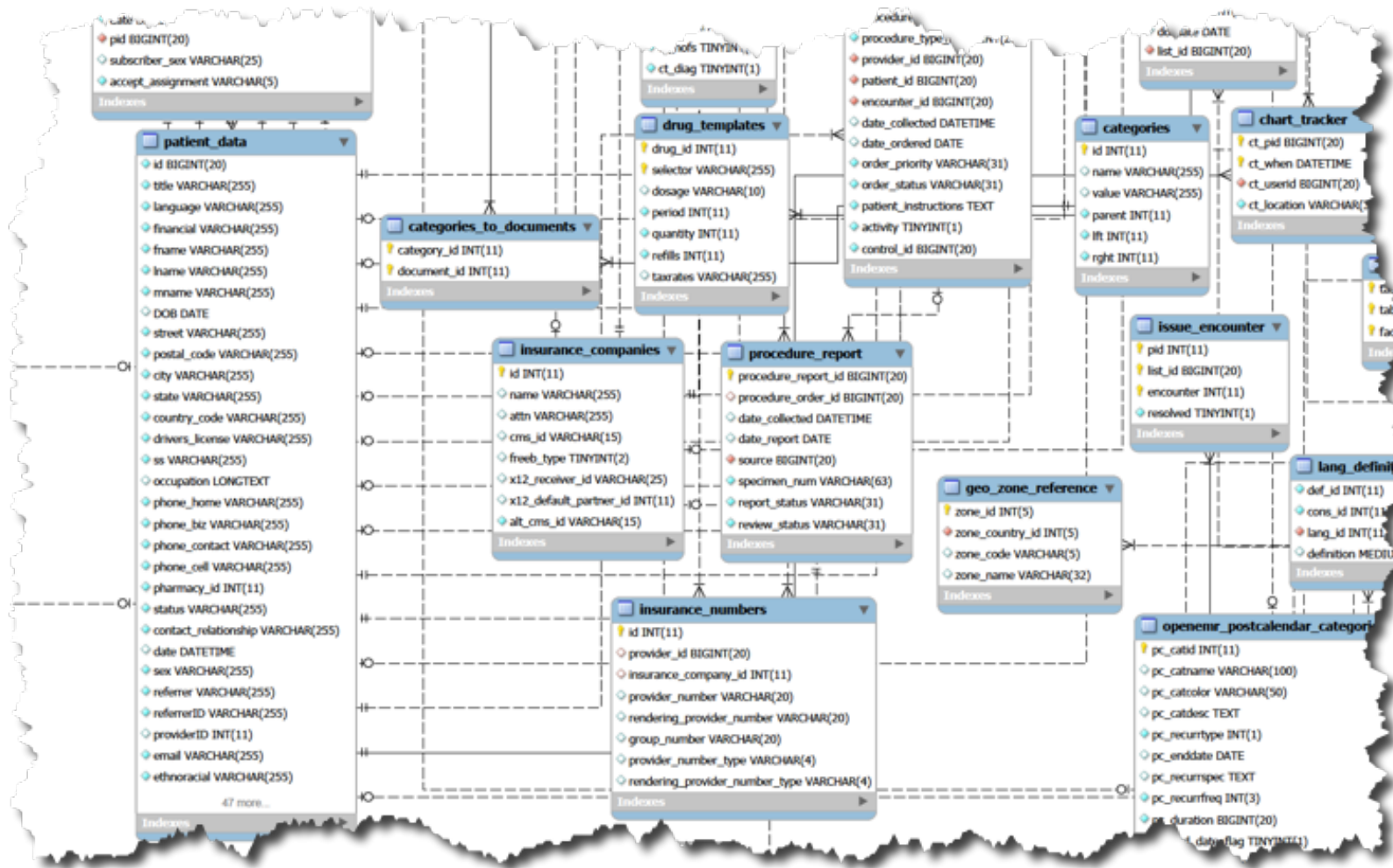
Indicates required data

Identifiers from Location, Provider, and Care Site tables

person	
P *	person_id
*	gender_concept_id
*	year_of_birth
	month_of_birth
	day_of_birth
	race_concept_id
	ethnicity_concept_id
F	location_id
F	provider_id
F	care_site_id
	person_source_value
	gender_source_value
	race_source_value
	ethnicity_source_value

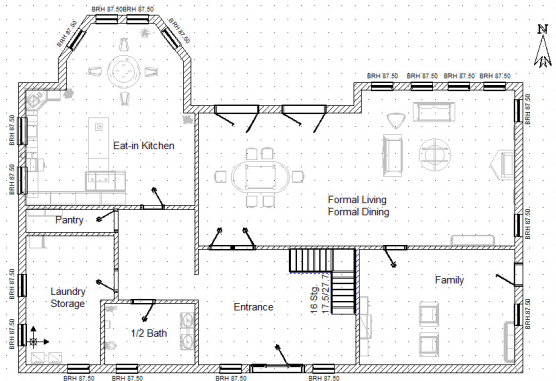
Attributes

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

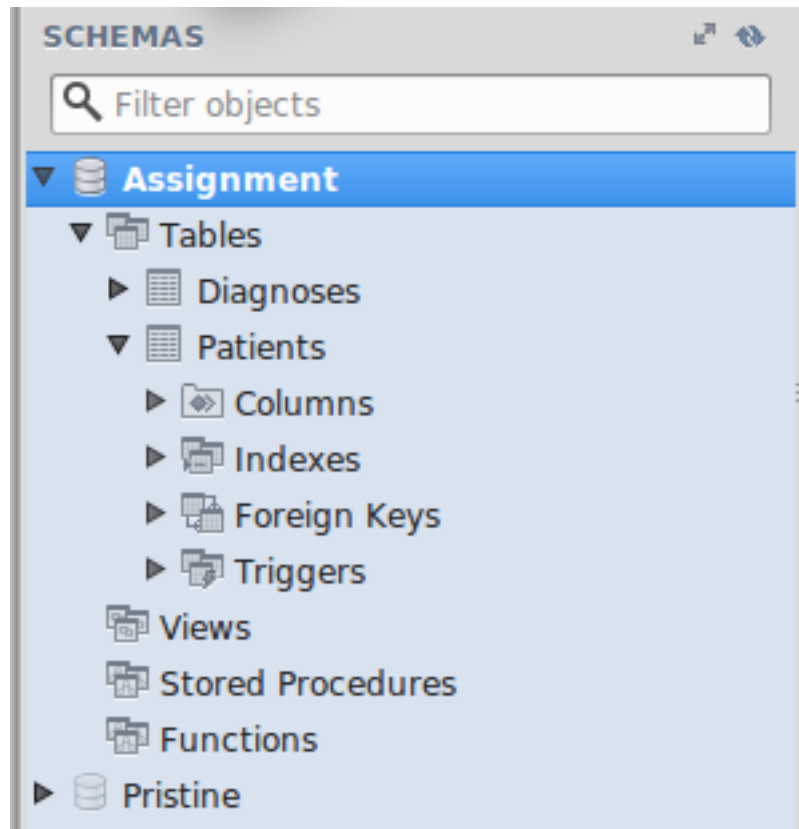
- **Database** – like the walls and foundation of a house
- **Tables** – like rooms for the data
- **Entries** (rows) in tables – each row is a specific patient, doctor, medication, etc.
- **Schema or Entity Relationship Diagrams (ERDs)** are like blueprints and floor plans – visualizations of the database 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 TABLENAME` gives you basic information

Explore the Patients table structure



Let's look at a table!

You Try It!

Run this command:

DESCRIBE Patients;

How does this compare to an Excel spreadsheet?

Selecting only specific columns

You Try It!

1. Select the last name, first name, and occupation for all patients
2. What is the occupation of the 14th patient on the list?

Answer

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

What is the occupation of the 14th patient on the list?

Answer

```
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?

PatientName	PatientPhone	DoctorName	DoctorPhone
Doe, John	123-456-7890	Smith, K	512-500-3920
Smith, Mary	123-654-0987	Jones, T	512-500-3921
Jones, Jill	123-321-2345	Smith, K	512-500-3920
Johnson, Scott	123-456-6789	Jones, T	512-500-3921
Baker, Tom	123-345-5678	Garcia, C	512-500-3923

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?

Answer

```
SELECT *  
FROM Patients;
```

2. How many patients are on the list?

3. What is the first name of the first patient on the list?

Answer

1. `SELECT *`
`FROM Patients;`

2. 790

3. What is the first name of the first patient on the list?

Answer

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)



`SELECT *` is
your friend!!

Adding Selection Criteria (Filters) with WHERE

Filtering the Selection

Patients Table				
ID	LastName	FirstName	StreetAddress	City
1	Doe	John	123 Main Street	Houston
2	Jones	Mary	425 Allison Drive	Austin
3	Smith	Erica	327 Maple Drive	Austin

- 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

The WHERE Clause, 1

Patients Table				
ID	LastName	FirstName	StreetAddress	City
1	Doe	John	123 Main Street	Houston
2	Jones	Mary	425 Allison Drive	Austin
3	Smith	Erica	327 Maple Drive	Austin

- Syntax:

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

The WHERE Clause, 2

```
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?

Answer

```
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?

Answer

```
SELECT * FROM Patients  
WHERE Gender LIKE 'M';
```

2. How many male patients are in the Patients table?

Answer

```
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 the number of patients born in 1959?
2. How many patients were born in 1959?

Answer

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

How many patients were born
in 1959?

Answer

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

12

You Try It!

1. What query would give you the 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)?

Answer

```
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

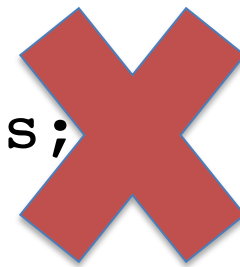
Consider this table – there are 6 patients with 3 different last names – Doe, Jones, and Smith

Patients Table				
ID	LastName	FirstName	StreetAddress	City
1	Doe	John	123 Main Street	Houston
2	Jones	Mary	425 Allison Drive	Austin
3	Smith	Erica	327 Maple Drive	Austin
4	Jones	Marcus	54 Congress Ave.	Austin
5	Smith	Alan	26 Bluebell Court	Austin
6	Smith	Donald	874 Elizabeth Ave.	Austin

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

You Try It!

1. What query would give you the number of unique states in the Patients table?
2. How many states are in the Patients table?

Answer

```
SELECT DISTINCT State  
FROM Patients;
```

How many states are in the
Patients table?

Answer

```
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;
```

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 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)

TxHiMA

<AHIMA Affiliate

Texas Health Information
Management Association ⁷⁹

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

```
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?

Answer

```
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

85

You Try It!

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?

Answer

```
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:

```
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

TxHiMA

— AHIMA Affiliate

Texas Health Information
Management Association

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

You Try It!

1. What query would give you the number of patients in the Patients table?
2. What number did you get?

Answer

```
SELECT COUNT( * )  
FROM Patients;
```

2. What number did you get?

Answer

```
SELECT COUNT( * )  
FROM Patients;
```

790

A little more difficulty...

- Let's try adding a WHERE clause!

You Try It!

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  
WHERE Gender 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

You Try It!

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?

Answer

```
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?

Answer

```
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

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



Notice the
parentheses

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!

You Try It!

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?

Answer

```
SELECT *  
FROM Patients  
WHERE Pounds LIKE  
'110.40';
```

2. What patient information did you get?

Answer

```
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

```
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?

Answer

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

2. What number did you get?

Answer

```
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?

CAUTION!

- Now run this query:

```
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:

```
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!

Action Output		Time	Action	Message
✓	1	10:06:47	SELECT * FROM Patients WHERE BloodType LIKE 'A+' LIMIT 0, 1000	190 row(s) returned
✓	2	10:14:44	SELECT * FROM Patients WHERE BloodType LIKE 'A-' LIMIT 0, 1000	23 row(s) returned
✓	3	10:14:47	SELECT * FROM Patients WHERE BloodType LIKE 'B+' LIMIT 0, 1000	193 row(s) returned
✓	4	10:14:52	SELECT * FROM Patients WHERE BloodType LIKE 'B-' LIMIT 0, 1000	10 row(s) returned
✓	5	10:14:57	SELECT * FROM Patients WHERE BloodType LIKE 'AB+' LIMIT 0, 1000	40 row(s) returned
✓	6	10:15:03	SELECT * FROM Patients WHERE BloodType LIKE 'AB-' LIMIT 0, 1000	2 row(s) returned
✓	7	10:15:08	SELECT * FROM Patients WHERE BloodType LIKE 'O+' LIMIT 0, 1000	297 row(s) returned
✓	8	10:15:12	SELECT * FROM Patients WHERE BloodType LIKE 'O-' LIMIT 0, 1000	35 row(s) returned

So, you could add COUNT to your queries...

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

Result Set Filter:  Export:  Wrap Cell Content: 

#	BloodType	COUNT(BloodType)
1	A+	190

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 ← *optional*

GROUP BY columnname(s)

ORDER BY columnname(s); ← *optional*

You Try It!

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;
```

#	BloodType	COUNT(BloodType)
1	A+	190
2	A-	23
3	AB+	40
4	AB-	2
5	B+	193
6	B-	10
7	O+	297
8	O-	35

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;
```


You Try It!

```
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

#	BloodType	COUNT(BloodType)
1	AB-	2
2	B-	10
3	A-	23
4	O-	35
5	AB+	40
6	A+	190
7	B+	193
8	O+	297

You Try It!



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

Result Set Filter:  Export:  Write

#	BloodType	COUNT(BloodType)
1	O+	297
2	B+	193
3	A+	190
4	AB+	40
5	O-	35
6	A-	23
7	B-	10
8	AB-	2

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 ← *optional*

GROUP BY columnname(s) ← *mandatory
for HAVING to work*

HAVING condition

ORDER BY columnname(s); ← *optional*

You Try It!

1. Run this:

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

2. What output did you get?

Answer

1. Run this:

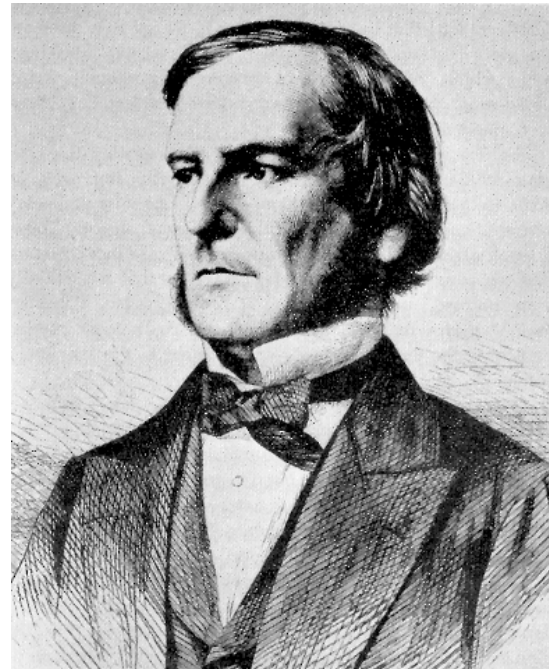
```
SELECT BloodType, COUNT(BloodType)
FROM Patients
GROUP BY BloodType
HAVING COUNT(BloodType) > 100;
```

#	BloodType	COUNT(BloodType)
1	A+	190
2	B+	193
3	O+	297

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



152

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

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 \geq 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 \geq 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 ≥ 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 ≥ 21 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 ≥ 21 years old AND have a blood type of A+.

$\geq 21?$	Has blood type A+?

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+.

$\geq 21?$	Has blood type A+?
No	No
No	Yes

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+.

$\geq 21?$	Has blood type A+?
No	No
No	Yes
Yes	No
Yes	Yes

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+.

$\geq 21?$	Has blood type A+?	Can participate?
No	No	
No	Yes	
Yes	No	
Yes	Yes	

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+.

$\geq 21?$	Has blood type A+?	Can participate?
No	No	No
No	Yes	No
Yes	No	
Yes	Yes	

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+.

$\geq 21?$	Has blood type A+?	Can participate?
No	No	No
No	Yes	No
Yes	No	No
Yes	Yes	

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+.

≥ 21 ?	Has blood type A+?	Can participate?
No	No	No
No	Yes	No
Yes	No	No
Yes	Yes	Yes

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+. ***Both conditions must be satisfied, or the record won't appear in our list of eligible patients.***

$\geq 21?$	Has blood type A+?	Can participate?
No	No	No
No	Yes	No
Yes	No	No
Yes	Yes	Yes

Truth Tables

- Mathematical table used to tell whether an expression is true for all legitimate input values
 - http://en.wikipedia.org/wiki/Truth_table
- 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

You Try It!

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?

Answer

```
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?

Answer

```
SELECT * FROM Patients  
WHERE Gender LIKE 'M' AND  
Pounds < 150  
ORDER BY FeetInches DESC;
```

5154

A Little Bit More Difficult

You Try It!

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

```
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-' );
```

181

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+' ;
```

You Try It!

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?

Answer

```
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 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:

```
SELECT * FROM Patients
```

```
WHERE LastName LIKE 'Smith' OR  
LastName LIKE 'Jones';
```

CAUTION!

Run this query:

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

How many records did you get?

CAUTION!

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

TxHiMA

← AHIMA Affiliate

Texas Health Information
Management Association

Putting Operators Together

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

Condition X	Condition Y	NOT X	NOT Y	(NOT X) OR (NOT Y)
0	0	1	1	1
0	1	1	0	1
1	0	0	1	1
1	1	0	0	0

195

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!

Much Better!

```
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

ID	Last_name	First_name	Street_Address	City	Dr_Iname	Dr_fname
1	Doe	John	123 Main Street	Houston	Jones	John
2	Jones	Mary	425 Allison Dr	Austin	Garcia	Lupe
3	Smith	Erica	327 Maple Drive	Austin	Gordon	Bill
4	Doe	John	123 Main Street	Houston	Gordon	Bill

- 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?

How Database Designers Fix This, 1

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

ID	Dr_Iname	Dr_fname
1	Jones	John
2	Garcia	Lupe
3	Gordon	Bill

ID	Last_name	First_name	Street_Address	City
1	Doe	John	123 Main Street	Houston
2	Jones	Mary	425 Allison Drive	Austin
3	Smith	Erica	327 Maple Drive	Austin
4	Doe	John	123 Main Street	Houston

How Database Designers Fix This,

2

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

ID	Dr_Iname	Dr_fname
1	Jones	John
2	Garcia	Lupe
3	Gordon	Bill

ID	Last_name	First_name	Street_Address	City	PrimaryMD	SecondaryMD
1	Doe	John	123 Main Street	Houston	1	3
2	Jones	Mary	425 Allison Drive	Austin	2	
3	Smith	Erica	327 Maple Drive	Austin	3	

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.



Photo by Octobergirl
<https://flic.kr/p/ktbSNc>
Creative Commons license

PATIENT
<u>Med_rec_number</u>
First_name
Last_name
Date_of_birth
Street_address
City
State
Zip
Driver_license_number
Social_security_number
Phone_number_home
Phone_number_work

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 ...

ID	Dr_First_name	Dr_Last_name
1	John	Jones
2	Lupe	Garcia
3	Bill	Gordon

... is inserted as a Foreign Key in the Primary Doctor column in the Patient table

ID	Last_name	First_name	Street_Address	City	Dr_Primary	Dr_Secondary
1	Doe	John	123 Main Street	Houston	1	3
2	Jones	Mary	425 Allison Drive	Austin	2	
3	Smith	Erica	327 Maple Drive	Austin	3	

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?

Answer

```
SELECT *  
FROM Diagnoses;
```

2. What can you tell about the first patient on the list?

Answer

```
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?

Answer

```
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

<i>Patients Table</i>				
MRNO	LastName	FirstName	MiddleInit	DOB
343	Curran	Barbara	N	1962-04-19

Is using these two fields
to link the tables together



<i>Diagnoses Table</i>			
ID	MRNO	DX1	Description 1
1	343	W59.21XA	Bitten by turtle, initial encounter

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

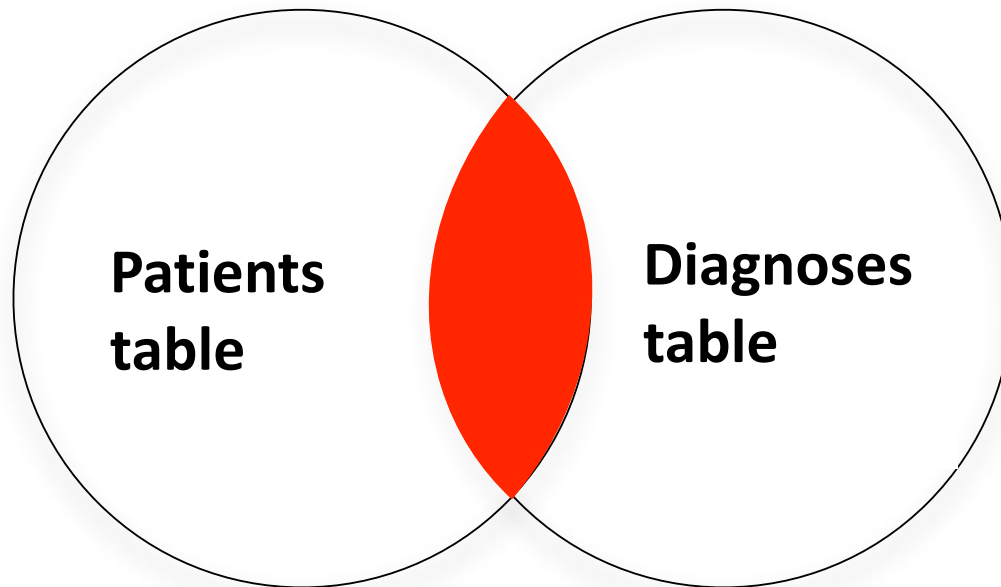
```
SELECT *  
FROM Patients JOIN Diagnoses  
on Patients.MRNO=Diagnoses.MRNO  
WHERE MRNO LIKE '343';
```


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?

Answer

```
SELECT * FROM Patients  
JOIN Diagnoses ON  
Patients.MRNO=Diagnoses.MRNO  
WHERE DX1 LIKE 'G40.901';
```

2. How many patients did you get?

Answer

```
SELECT * FROM Patients  
JOIN Diagnoses ON  
Patients.MRNO=Diagnoses.MRNO  
WHERE DX1 LIKE 'G40.901';
```

8

You Try It!

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?

Answer

```
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)

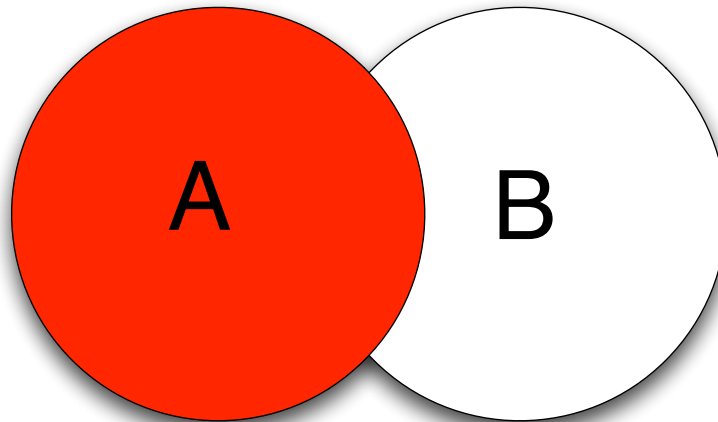
LEFT JOIN

- 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:

```
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?

Answer

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

790

JOINS - Summary

LEFT JOIN	RIGHT JOIN
Gives all records from Table A, with any matching records from Table B. If there is no match, the right side will be empty.	Gives all records from Table B, with any matching records from Table A. If there is no match, the left side will be empty.

RIGHT JOIN

(sometimes also called Right Outer Join)

RIGHT JOIN

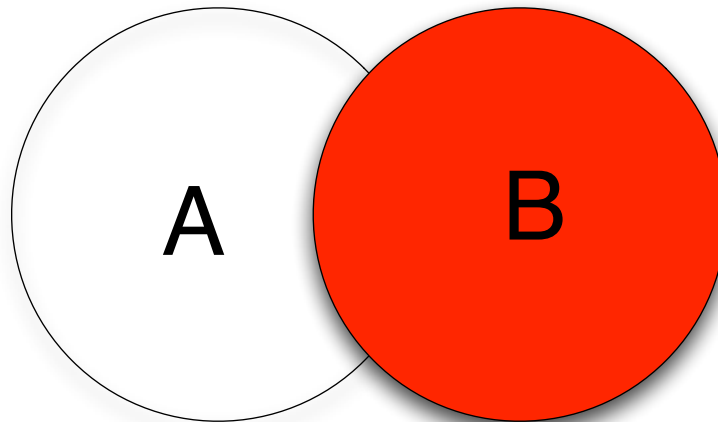
- RIGHT JOIN returns all records from the right table (table_name2), even if there are no matches in the left table (table_name1)
- If there are no matching records from table_name1, then those fields will be empty
- Syntax:

```
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?

Answer

```
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!

Occupation	Employer	BloodType	Pounds	FeetInches	AdmitDate	ID	MRNO	DX1	Description
Electrical and electronics engineering technician	L.L. Berger	O+	228.40	5' 1"	2018-05-08	41	7136	E11.9	Type 2 d
Agricultural product grader	Integra Wealth	O+	535.60	5' 5"	2018-05-09	42	8345	E11.9	Type 2 d
Lathe and turning machine tool operator	The Pink Pig Ta	O+	543.80	6' 3"	2018-05-10	43	9589	E11.9	Type 2 d
NULL	NULL	NULL	NULL	NULL	NULL	44	326	E10.9	Type 1 d
NULL	NULL	NULL	NULL	NULL	NULL	45	1528	E10.9	Type 1 d
Crushing grinding and polishing machine operator	MegaSolutions	A+	243.80	5' 9"	2018-05-03	46	2632	E10.9	Type 1 d
Middle school teacher	Edge Garden S	A+	205.90	6' 0"	2018-05-05	47	4045	E10.9	Type 1 d

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)

Name	Description	Example
char	Character string	John Doe
Int	Integer	Whole number, can be positive or negative
bool	Boolean	True or False
float	Floating point number	Numbers with decimal points

240

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

The image shows a screenshot of an online form with several fields. Blue callout boxes with arrows point to specific fields, indicating their data types:

- Char or string:** Points to the First Name field.
- Date:** Points to the Date of Birth field.
- Boolean:** Points to the Gender field (Male/Female radio buttons).
- Might be integer or character:** Points to the MRN field.

The form fields are:

- * First Name:
- Middle Name:
- * Last Name:
- Suffix:
- * Date of Birth: mm/dd/yyyy
- * Gender: Male Female
- Social Security #: XXX-XX-
- MRN:
- * e-Mail Address:

Let's create a Providers table

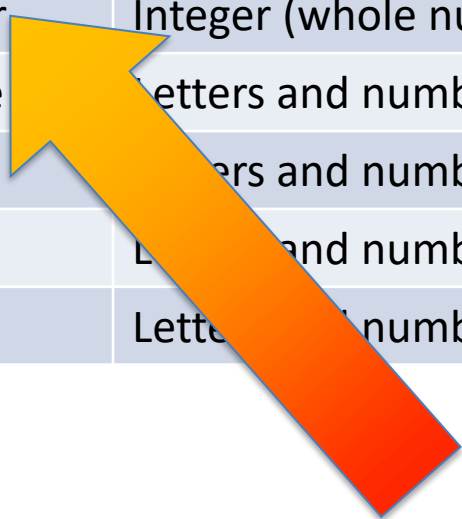
- Table name: Providers
- We want to store the following data elements:
 - ID number
 - First name
 - Last name
 - Specialty
 - Phone number

Defining Data Elements

Data Element	Data Type	How many characters?
ID number	Integer (whole numbers only)	20
First name	Letters and numbers (“varchar”)	50
Last name	Letters and numbers (“varchar”)	50
Specialty	Letters and numbers (“varchar”)	50
Phone	Letters and numbers (“varchar”)	15

What will be the primary key?

Data Element	Data Type	How many characters?
ID number	Integer (whole numbers only)	20
First name	Letters and numbers ("varchar")	50
Last name	Letters and numbers ("varchar")	50
Specialty	Letters and numbers ("varchar")	50
Phone	Letters and numbers ("varchar")	15



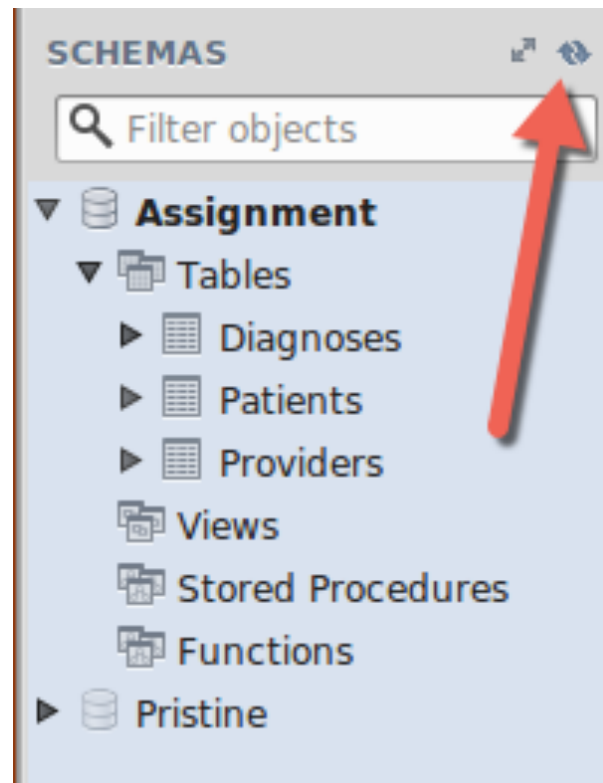
You Try It!

```
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:

INSERT INTO tablename

(column1, column2
columnN)

VALUES

('value1', 'value2'.... 'valueN');

You Try It!

```
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!

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

Answer

```
SELECT * FROM Providers;
```

#	DrIDNumber	DrFirstName	DrLastName	DrSpecialty	DrPhone
1	24852	Frank	Wagoner	Obstetrics	713-555-3572
2	75623	Loren	Morgan	Nephrology	713-555-3465
*	NULL	NULL	NULL	NULL	NULL

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.

Answer

```
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

```
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)

You Try It!

```
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?

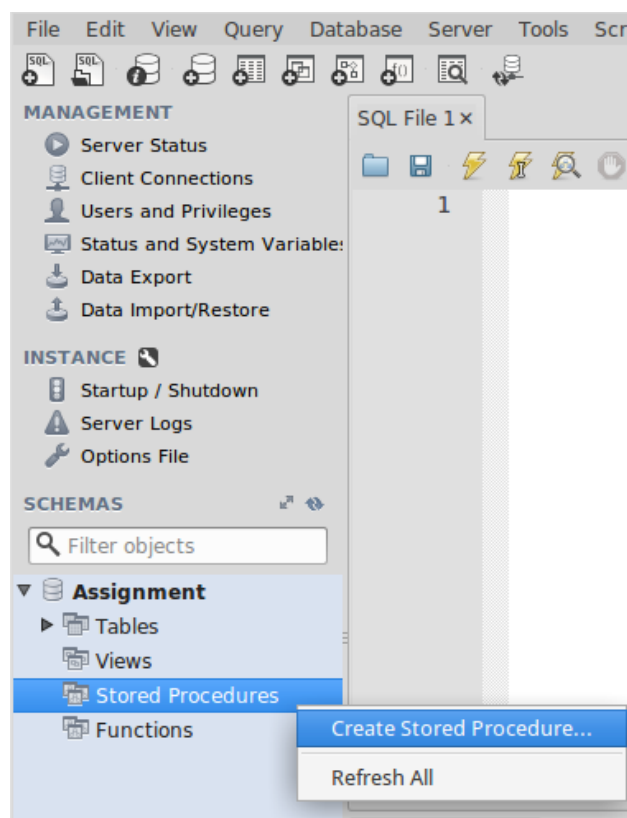
List of additional training resources is on my web page

Thank you!

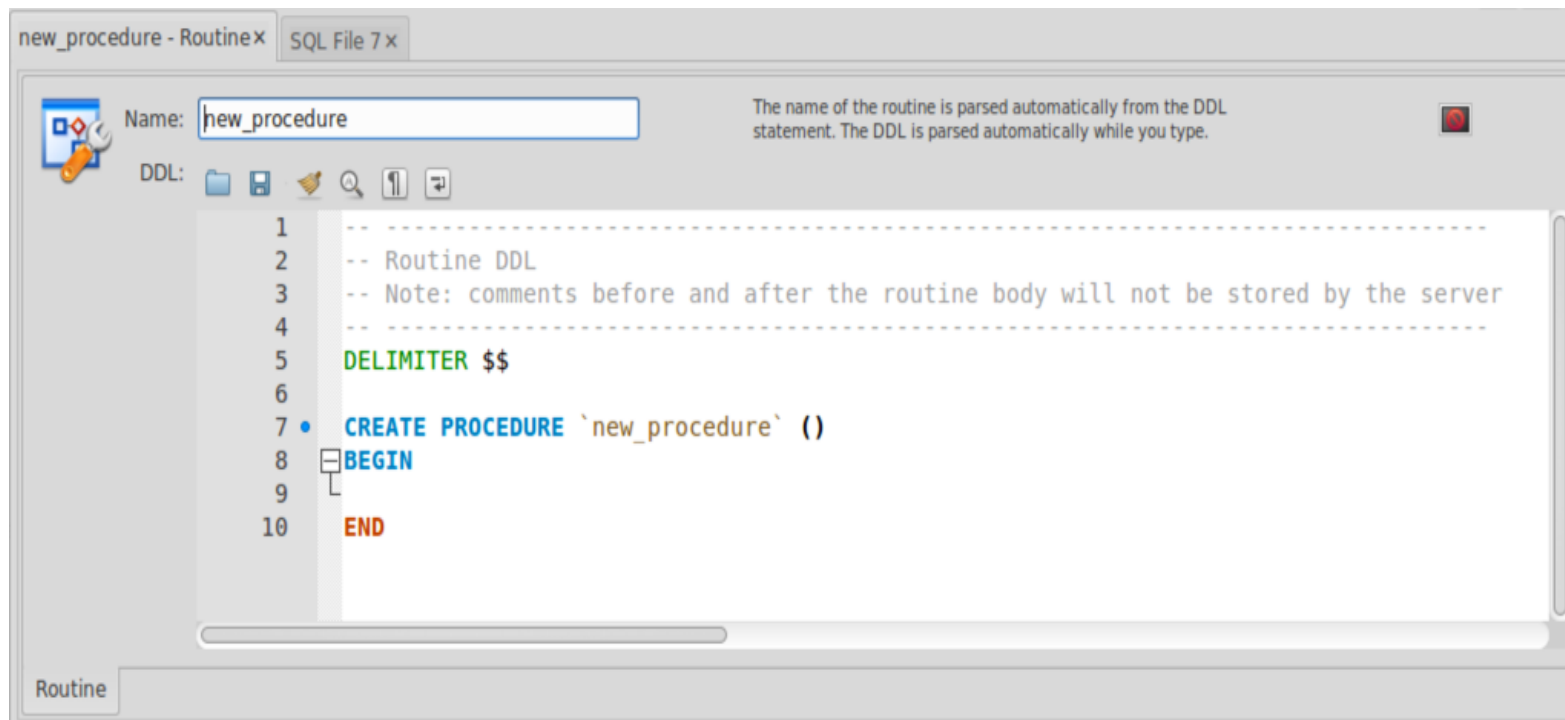
If we have time: Stored Procedures

What is a stored procedure?

- Creating a stored procedure



The stored procedure definition screen

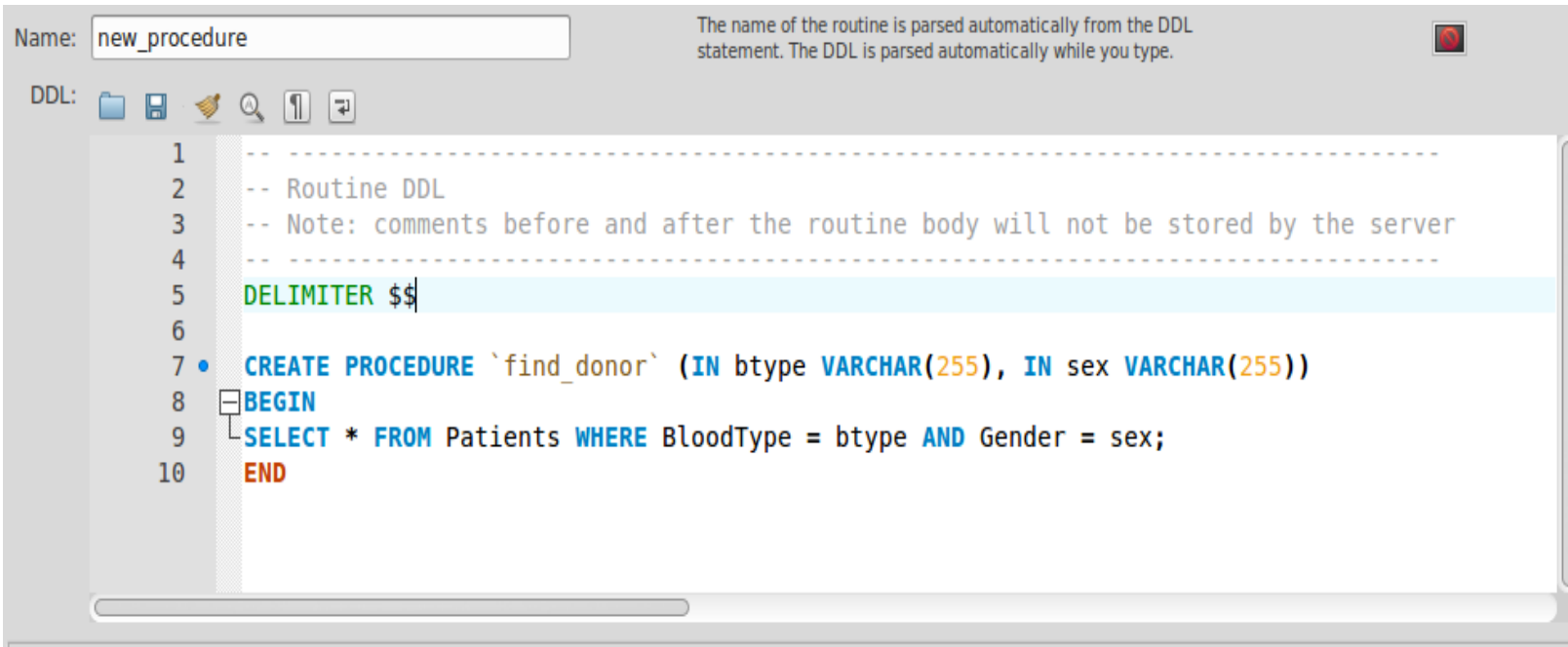


Under Delimiter \$\$:

```
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

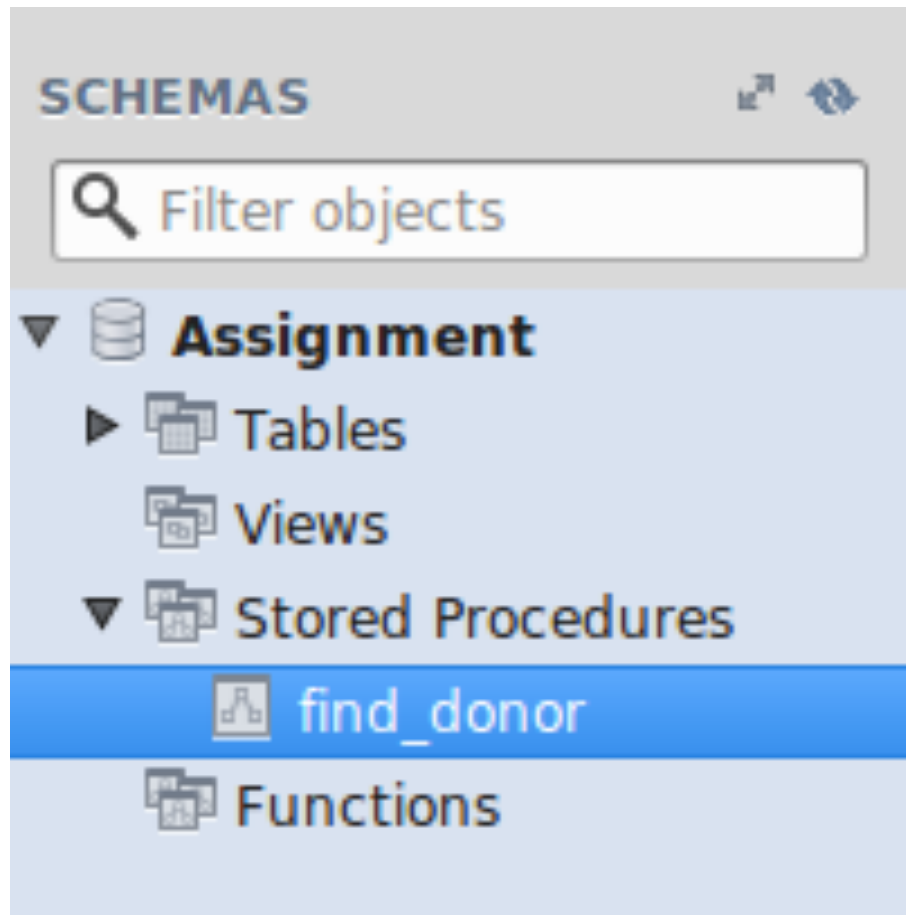
- Click Apply



The screenshot shows a database IDE interface. At the top, there is a text box labeled 'Name:' containing 'new_procedure'. To its right, a message states: 'The name of the routine is parsed automatically from the DDL statement. The DDL is parsed automatically while you type.' Below this is a toolbar with icons for file operations. The main area is a code editor with a line number column on the left (1-10). The code is as follows:

```
1  -----
2  -- Routine DDL
3  -- Note: comments before and after the routine body will not be stored by the server
4  -----
5  DELIMITER $$
6
7  • CREATE PROCEDURE `find_donor` (IN btype VARCHAR(255), IN sex VARCHAR(255))
8  BEGIN
9  SELECT * FROM Patients WHERE BloodType = btype AND Gender = sex;
10 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