Data Analytics: Introduction to SQL using Healthcare Data

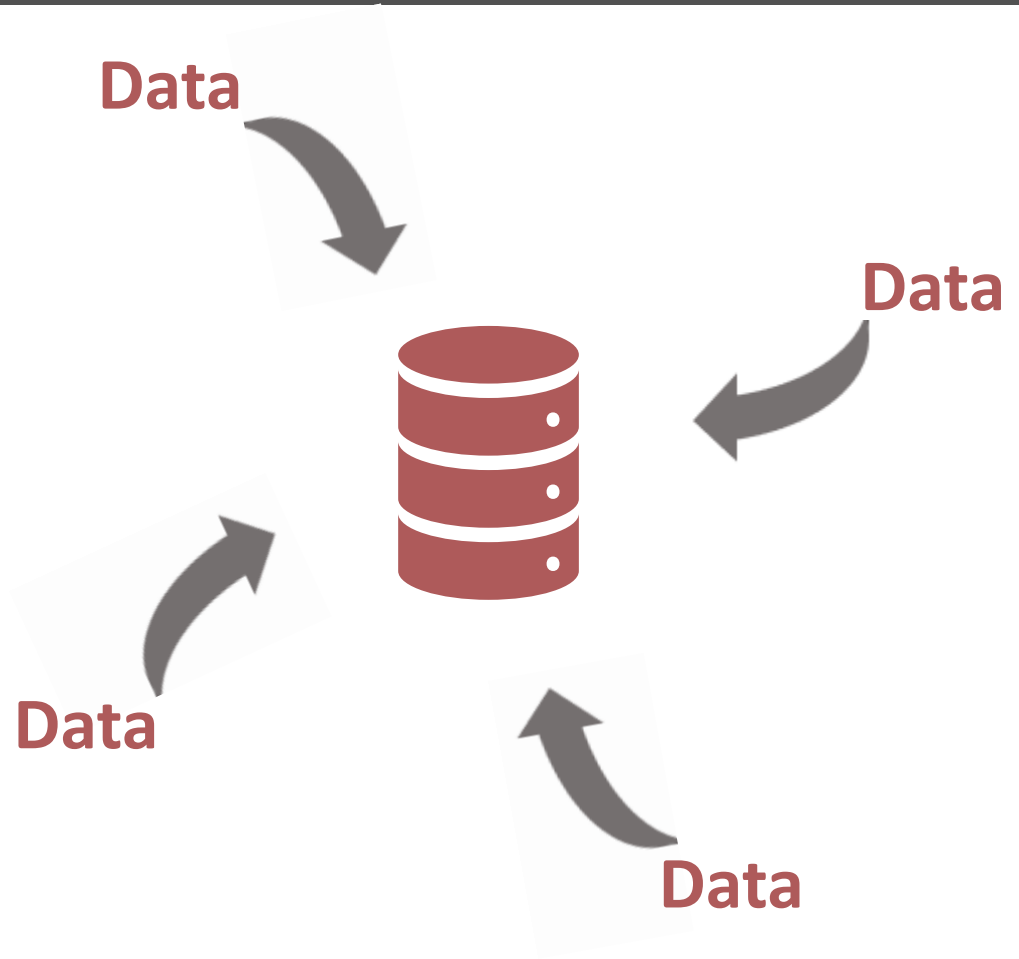
Microsoft SQL Server Management

*A SQL Course designed **specifically** for aspiring data professionals to learn **Microsoft SQL Server***

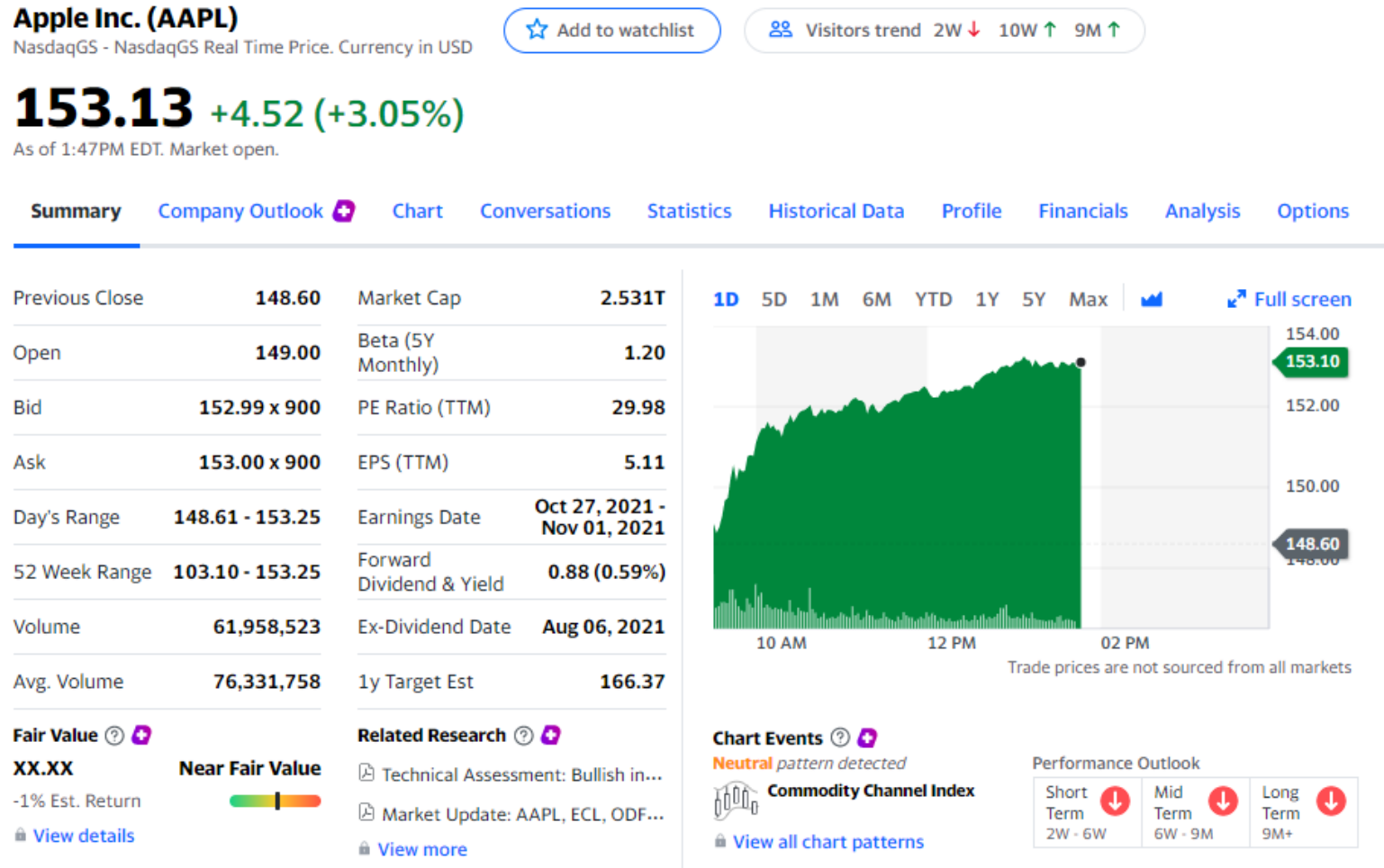
Section 1: Introductions

- *What you will learn in this course*
- *Introduction to SQL*
- *Quick Glimpse into the data used in this course*
- *Cheat Sheet*

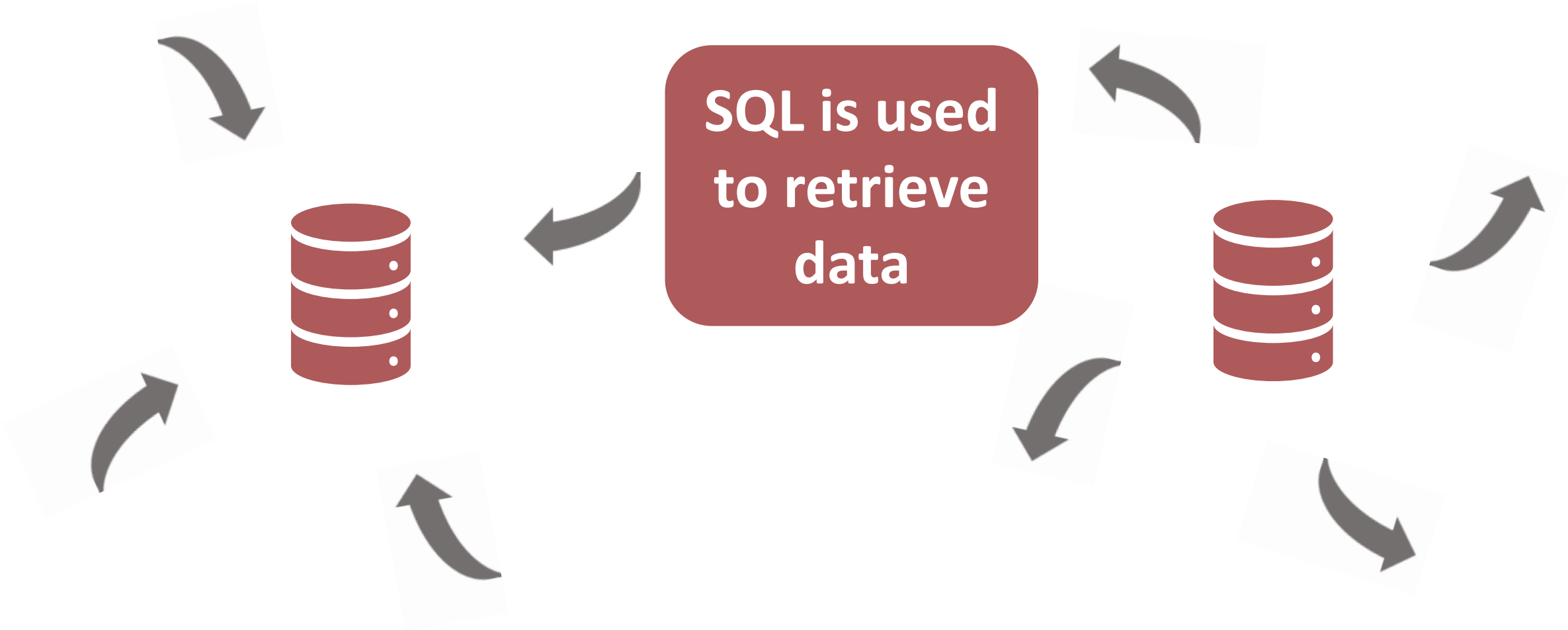
What is a Database?



Database example



SQL is an abbreviation for Structured Query Language. In other words, SQL is a language used to communicate with databases. Using SQL users can update, retrieve, delete, create data.

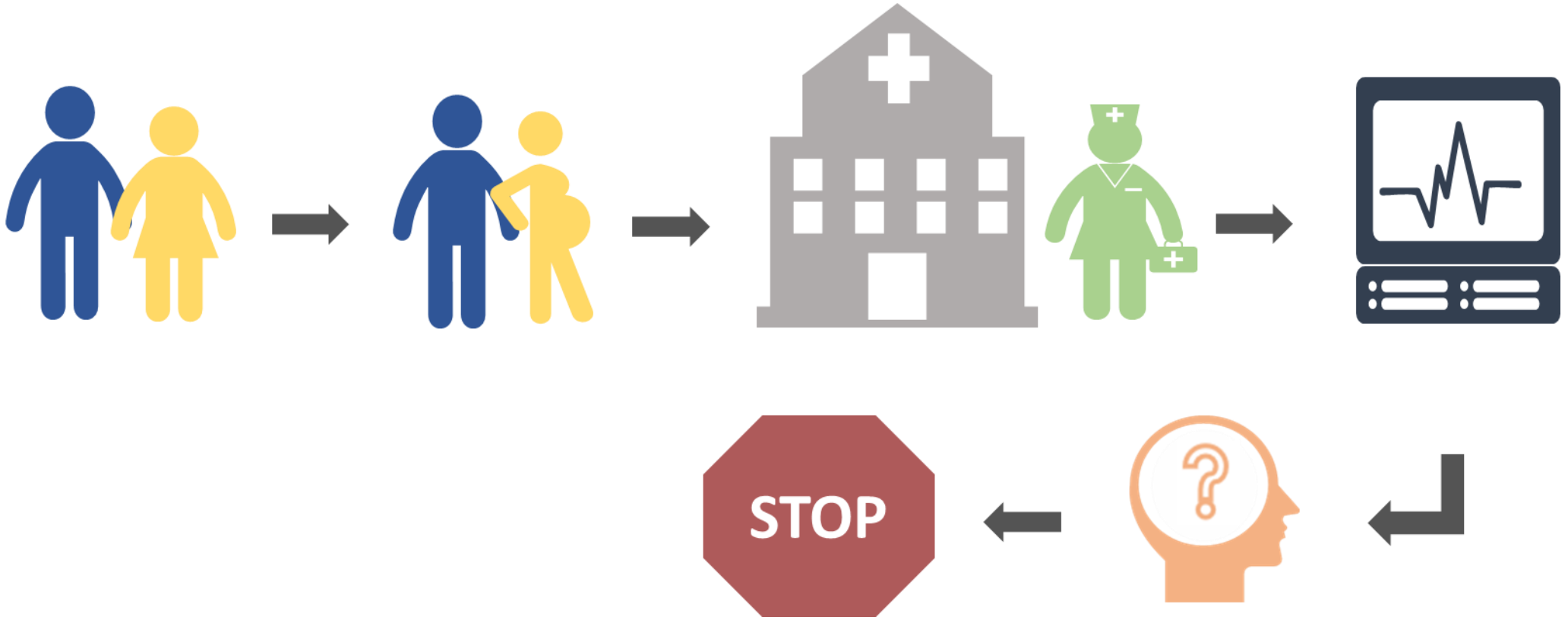

SQL is used to retrieve data

There are different types of database management systems. The SQL used in these systems vary slightly, but for the most part are very similar

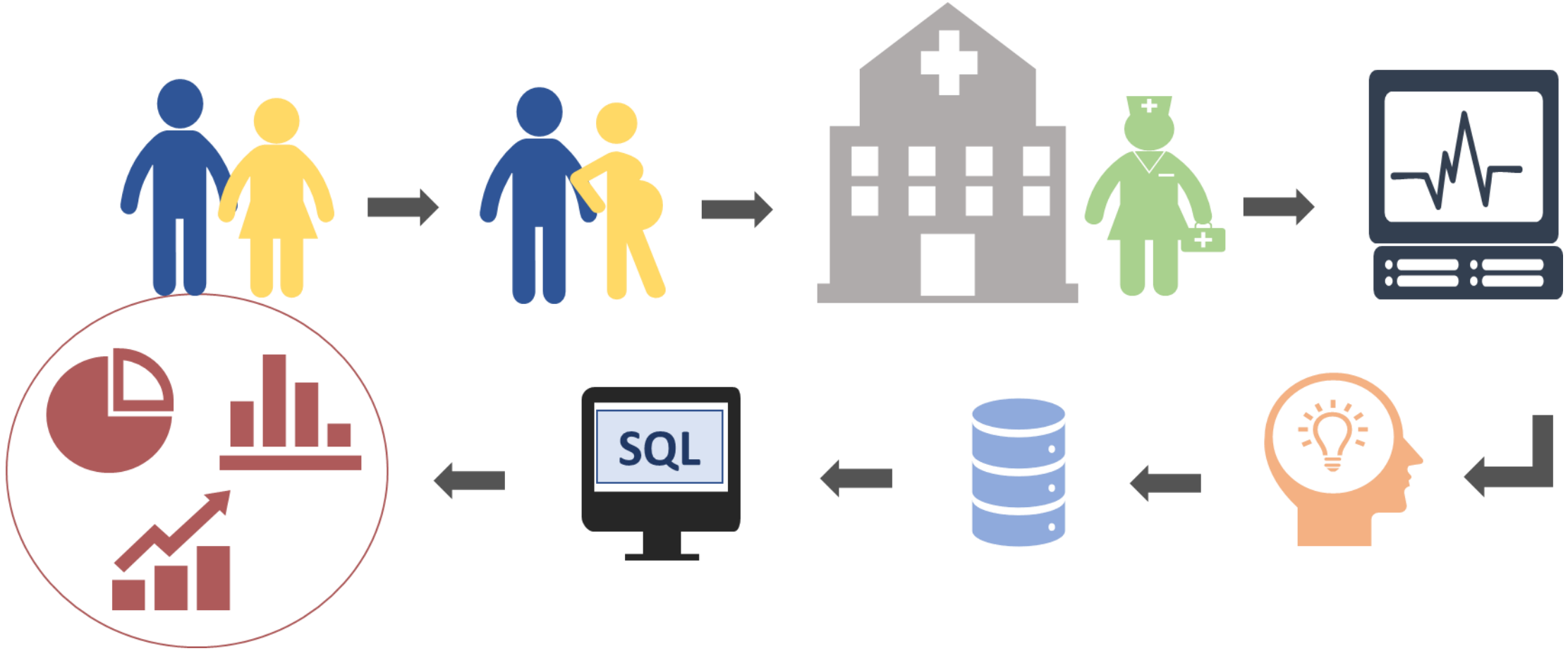
ORACLE
DATABASE

SYBASE
An SAP Company

Section 1: The Healthcare Data used in this course



Section 1: The Healthcare Data used in this course



Section 1: Cheat Sheet

Basic Syntax

```
SELECT * FROM table_name
  -> Populates the whole table
SELECT column1, col2, col3… FROM table_name
  -> Populates specified columns
WHERE col2 = condition (=,>,<,>=,<=)
  -> filter rows where column values meet
condition
GROUP BY col1, col3
  -> Groups rows that have the same values
HAVING Count(*) > value
  -> Limit Aggregated Data
ORDER BY col4 (DESC or ASC)
  -> Order you results by a column
```

Useful Keywords when using SELECTS

```
DISTINCT -> Returns unique rows
BETWEEN a AND b -> Limits range of values
LIKE -> Pattern Search within the column values
IN(a,b,c) -> Returns values contained among list
TOP 100 -> Select top number of rows
```

Aggregation Functions

```
COUNT -> Count of rows SUM -> Cumulates values
AVG -> Avg's Values Max/Min -> Small/large values
```

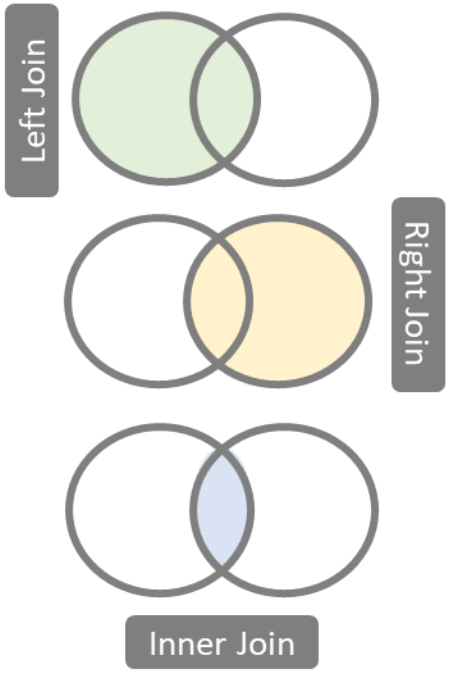
Table Manipulation

```
CREATE TABLE table_name (col1 datatype, col2 datatype…)
  -> Creates new table, specify the type of data in columns
DROP TABLE table_name
  -> Permanently deletes data table
TRUNCATE TABLE table_name
  -> Deletes data values in table, but table still exists
INSERT INTO table_name (col1, col2) VALUES (value1, value2)
  -> Insert data into created table
ALTER TABLE table_name ADD column_name datatype
  -> Add or Delete columns from table
UPDATE TABLE table_name SET col1 = value1, col2 = value2…
  -> Update existing records in a table
```

Joins

```
SELECT * FROM table1_name INNER JOIN table2_name ON
     table2_name.column1 = table1_name.column1
  -> Joining two tables using like columns
INNER JOIN -> Combing rows from tables where JOIN is true
LEFT JOIN -> Returns all records from left table and
matched records from the right table
RIGHT JOIN -> Returns all records from right table and
matched records from the left table
```

Joins Visualized



Section 2: Downloading Software

- *Downloading SQL Server Management Studio (SSMS)*
  - *Other SQL applications can be used*
  - *However, SSMS will be the application the instructor uses*

Section 2: Downloading Microsoft SQL Server

1. Click Link below
   - https://www.microsoft.com/en-us/sql-server/sql-server-downloads
2. Express – “Download Now”
3. Open .exe file in browser
4. User Control – Click “Yes”
5. Click “Basic”
6. License Terms – Click “Accept”
7. Install Location – “Install”
8. Wait (take a break)
9. Install SSMS
10. Redirect to website (scroll down)
11. Click “Download SQL Server Management Studio (S”
12. Open .exe file in browser
13. Install
14. Setup Completed

* Make sure computer is up-to-date.
** These steps are for windows 10.

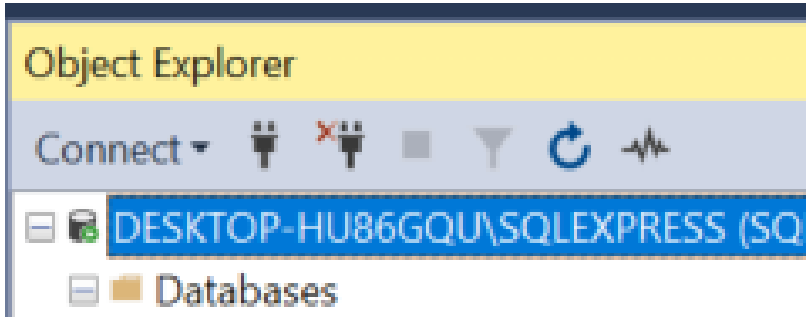
It is important to follow steps very closely. Unfortunately, this is an online course, which means helping you troubleshoot will be a challenge. There are many resources via the internet to help troubleshoot. Instructor will help where possible.

Section 3: Servers and Databases

- *Connecting to server in SSMS*
- *Creating Databases*
- *Dropping Databases*
- *Using Databases*

Section 3: Connecting to Sever

1. Open Microsoft SQL Server
2. Click Connect
3. Choose “Database Engine”
4. A window will pop up to input server name
5. The server name was created during download. Unless you changed anything the server name should be “SQLExpress”
6. Leave Authentication as “Windows Authentication”
7. Click “Connect”
8. Boom! Let's go.



```
Create Database DatabaseName

-> Create Database SQLCourse_DB
```

-> Example Statement

```
Drop Database DatabaseName
```

```
-> Drop Database SQLCourse_DB
```

-> Example Statement

```
Use DatabaseName
```

```
-> Use SQLCourse_DB
```

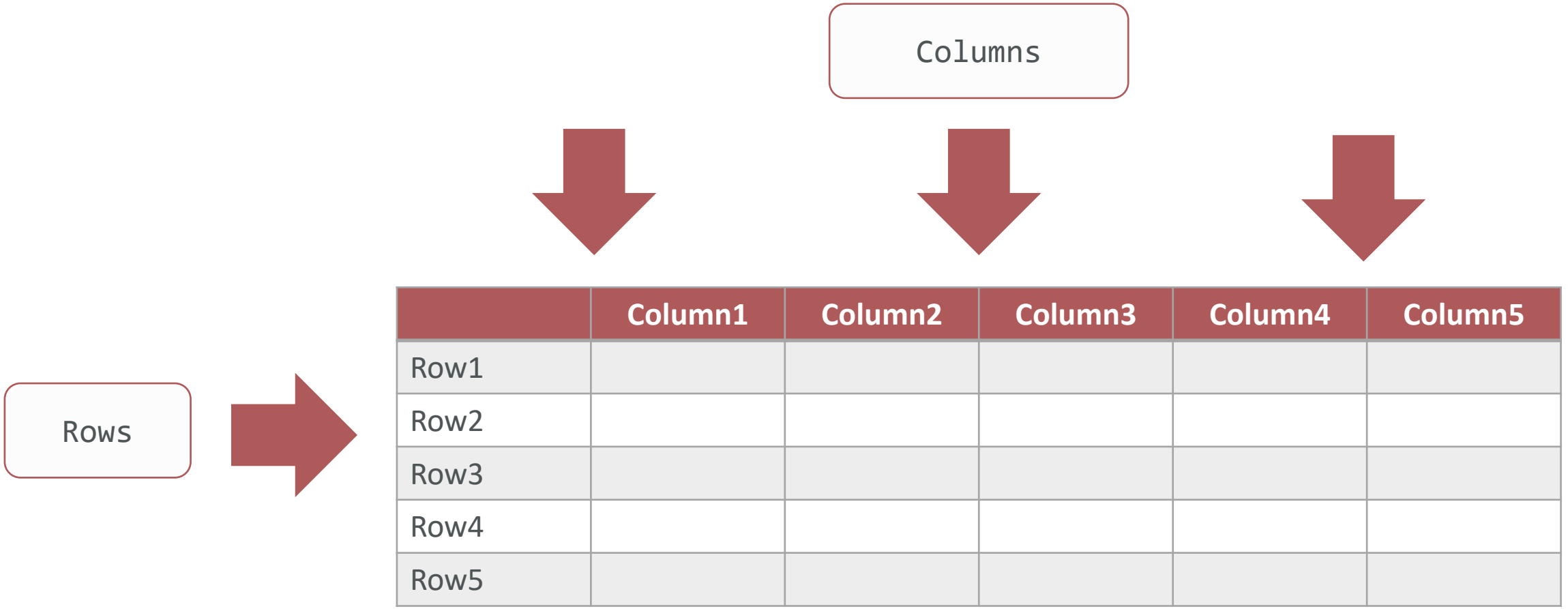
-> Example Statement

Section 3: Self-Evaluation

Step 1: Create a Database called “SQL_Course”
Step 2: Drop the same Database
Step 3: Create the same Database – “SQL_Course”
Step 4: Use the Database

Section 4: Creating and Inserting into tables

- *Table Structure*
- *Data Types*
- *Creating Tables*
- *Dropping Tables*
- *Inserting Data*
- *Tables with null values*
- *Creating Tables with Primary keys*
- *If Object_ID*



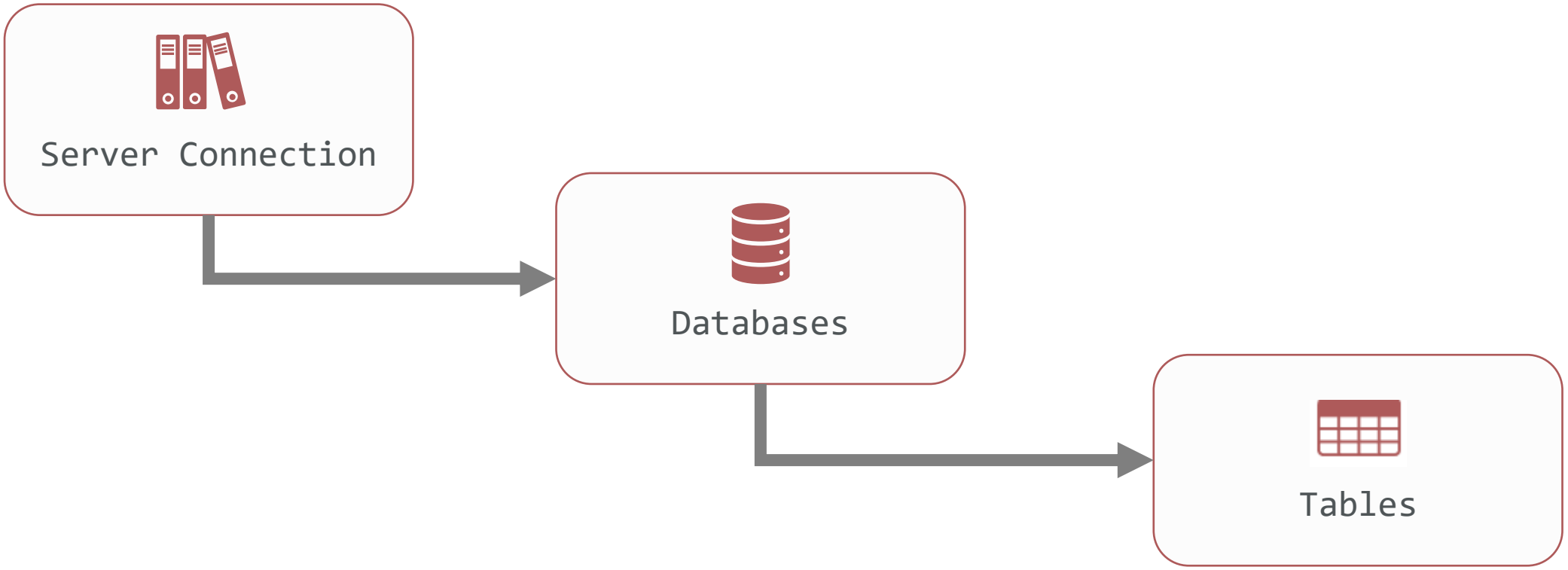
	Column1	Column2	Column3	Column4	Column5
Row1					
Row2					
Row3					
Row4					
Row5					

What is a datatype and why are they important?

Patient Name	Charges	Visits	Charge per visit
Bob	$500	2	$250
Jill	$3,000	4	$750
Jack	$5,000	Six	??!??
Dorothy	$5,000	8	$625

$3,000/4 = $750

Data Type	Description	Example
Int	Integers that are whole numbers. There are maximum and minimum values in SQL server (-/+ 2,147,483,648)	2,4,8,3000
Varchar	Variable text character. Must specify the number of characters	Hello, Text
Datetime	Stores date and time	YYYY-MM-DD H:MM:SS
Decimal	Decimal points and cannot have more than 38 digits	8.9


Server Connection
Databases
Tables

```
Create Table TableName
   (Column1 datatype
   ,Column2 datatype
   ,Column3 datatype…)

-> CREATE TABLE TestTable
   (PatientID varchar(255)
    ,PatientName varchar(255)
    ,PatientState varchar(255)
    ,Gender varchar(255)
    ,Visits int
    ,Charges int)
```

Section 4: Dropping Tables

```
Drop Table TableName
```

-> Drop TABLE TestTable

-> Example Statement

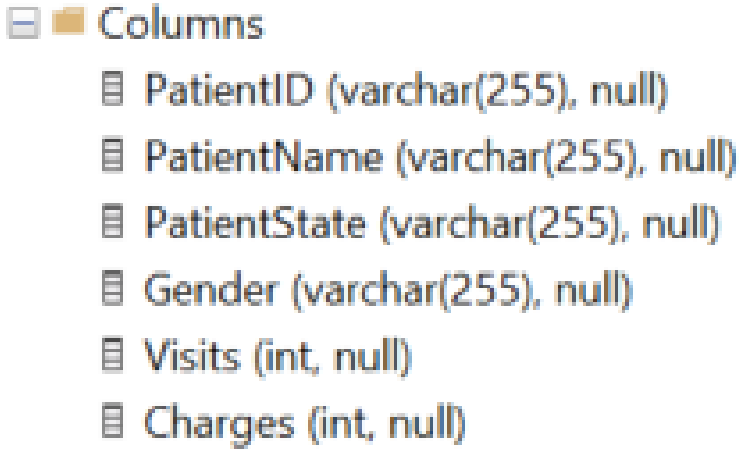
```
Insert Into TableName
  (column1, column2, column3, etc.)
Values (Value1, Value2, Value3, etc.)
```

```
-> INSERT INTO TestTable
(PatientID ,PatientName
,PatientState ,Gender ,Visits ,Charges)
VALUES ('12345','John','AL','M','3','200')
```

-> Example Statement

```
      -> INSERT INTO TestTable
(PatientID ,PatientName ,PatientState ,Gender ,Visits ,Charges)
          VALUES ('12345','John','AL','M','3','200’)
                ,('12346','Jane','AK','F','1','400’)
                ,('12347','Alex','AZ','F','6','900’)
                ,('12348','Bob','CA','M','7','8000’)
                ,('12349','Josh','CO','M','12','19000’)
                ,('12350','Stephanie','FL','F','18','25000’)
                ,('12351','Amber','GA','F','4','400’)
                ,('12352','Brittany','GA','F','6','4000’)
                ,('12353','Bill','UT','M','8','5000’)
                ,('12354','Nate','WY','M','22','28000')
```

-> Example Statement




```
->INSERT INTO TestTable
(PatientName)
VALUES ('Fred')
```

```
-> Select * from TestTable
```

-> Example Statement

Section 4: Null Values – Setting a default

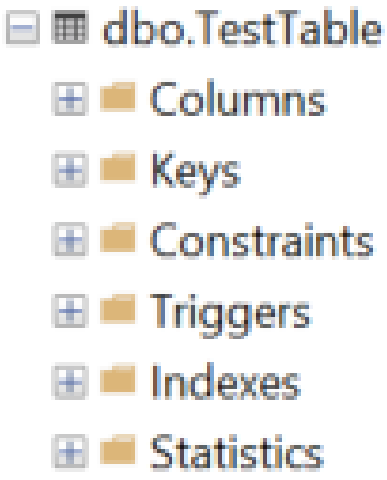
Null Value Reminders

1. A null value is not the same thing as a zero value
2. A Null value has been left blank
3. If you the field is optional, then a null value can be saved there
4. SQL allows default value rather than Null

-> Example Statement

```
-> CREATE TABLE TestTable
(PatientID varchar(255) NOT NULL,
PatientName varchar(255) NOT NULL,
PatientState varchar(255) NOT NULL,
Gender varchar(255) NOT NULL,
Visits int NULL
,Charges int NULL Default 0)
```

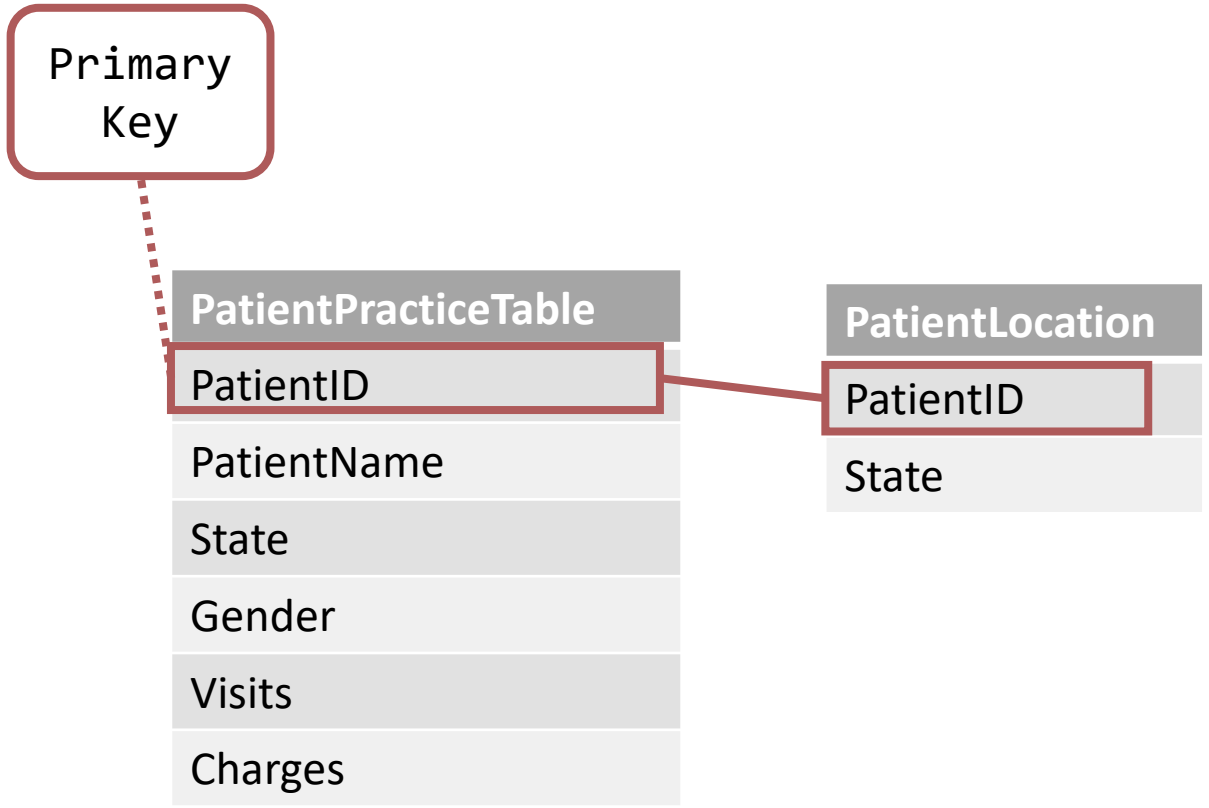
Rather than leaving the value as NULL we can create a default value, such as “0”



PatientID	PatientName	PatientSta	Gender	Visits	Charges
12355	Fred	CA	M	3	500
12355	Fred	CA	M	3	500
12355	Fred	CA	M	3	500
12355	Fred	CA	M	3	500
12355	Fred	CA	M	3	500
12355	Fred	CA	M	3	500

-> Example Statement

Section 4: Primary Keys



PatientID	PatientName	Gender	Visits	Charges
12345	John	M	3	$ 200
12346	Jane	F	1	$ 400
12347	Alex	F	6	$ 900
12348	Bob	M	7	$ 8,000
12349	Josh	M	12	$ 19,000
12350	Stephanie	F	18	$ 25,000
12351	Amber	F	4	$ 400
12352	Brittany	F	6	$ 4,000
12353	Bill	M	8	$ 5,000
12354	Nate	M	22	$ 28,000

PatientID	State
12345	AL
12346	AK
12347	AZ
12348	CA
12349	CO
12350	FL
12351	GA
12352	GA
12353	UT
12354	WY

PatientID	PatientName	State	Gender	Visits	Charges
12345	John	AL	M	3	$ 200

-> Example Statement

Primary Key Reminders

1. A Primary key must contain unique values
2. A table can only have one primary key
3. The primary key identifies each record in a table and can connect multiple tables together

-> Example Statement

```
-> CREATE TABLE TestTable
(PatientID int NOT NULL PRIMARY KEY,
PatientName varchar(255) NULL,
PatientState varchar(255) NULL,
Gender varchar(255) NULL,
Visits int NULL,
Charges int NULL Default 0)
```

```
   IF OBJECT_ID('TableName') IS NOT NULL DROP TABLE TableName
   GO

-> IF OBJECT_ID('TestTable') IS NOT NULL DROP TABLE TestTable
   GO
```

-> Example Statement

Section 4: Self Evaluation

Step 1: Use the database “SQL_Course” (created in previous self evaluation)
Step 2: Create a table called “SQL_CourseTable”
Step 3: This table will have 4 columns – “ID, Name, Address, Visits”
- DataTypes
  - ID – varchar(50)
  - Name – varchar(255)
  - Address – varchar(255)
  - Visits - int

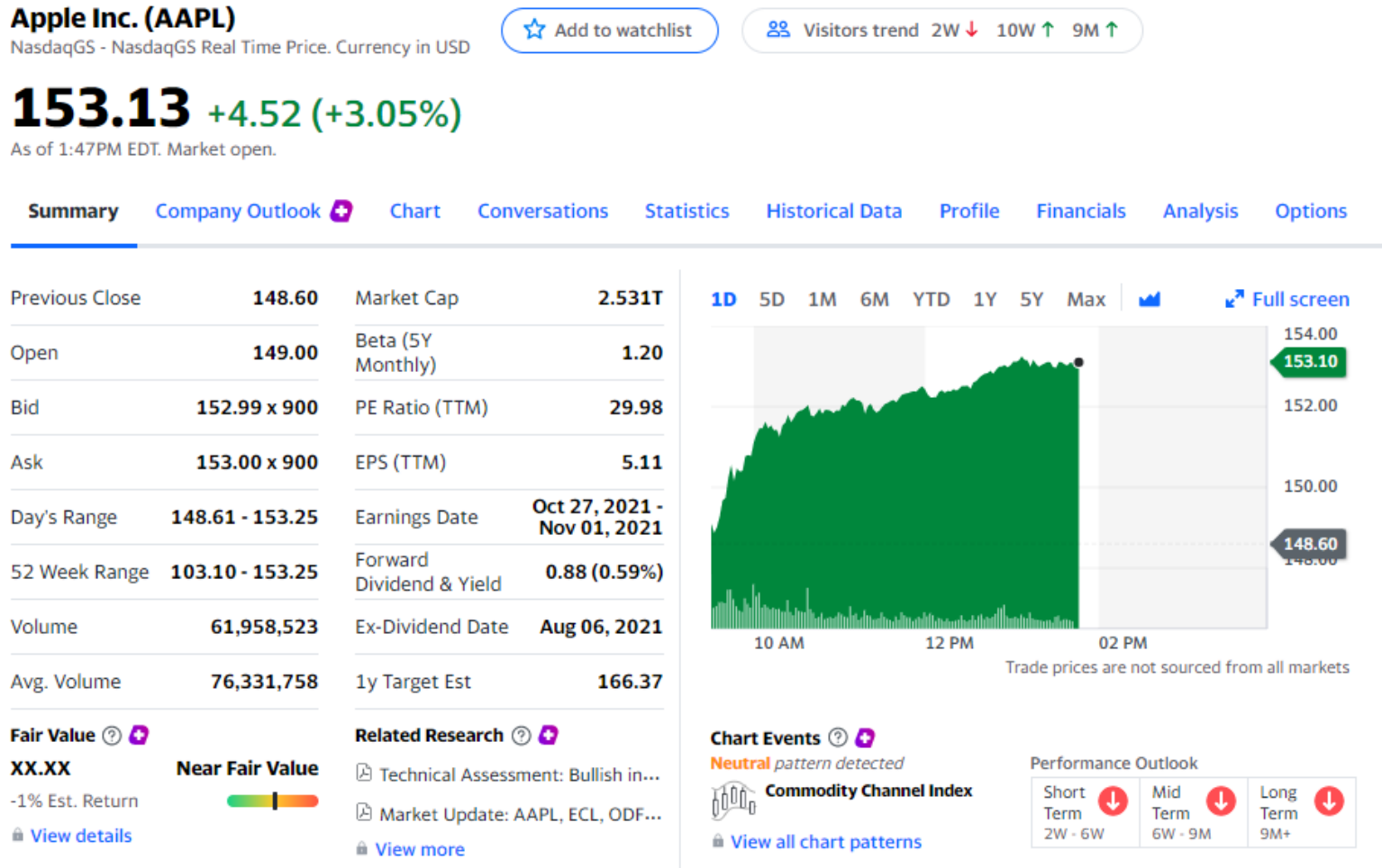
Step 4: Make ID a primary key
Step 5: ID cannot have null values
Step 6: If “Visits” has a null value then set default value to “0”
Step 7: Insert 5 rows of data (make up your own data)

```
-> Example Statement
```

Section 5: Retrieving Data

- *Select Statements*
  - *Top 100, Distinct*
- *Where Clause*
  - *And, Or, Not, Null, Not Null*
  - *Like, In, Between*
- *Common Wildcard symbols*
- *Order by*
- *Group by*
- *Aggregate Functions*
- *Alias Names*
  - *Concatenation*
- *Having*
- *Case When*

Database example



Section 5: Section Introduction

```sql
Select
PatientName
,PatientState
,Sum(Charges) as Charges
,Sum(Visits) as Visits
,Sum(Charges)/Sum(Visits) as AvgChargePerVisit
From TestTable
Where Gender = 'F'
and PatientState in ('GA','FL')
Group by
PatientName
,PatientState
Having Sum(Charges) >= 900
Order by PatientName
```

```sql
Select
From
Where
Group by
Having
Order by
```

```
Select * From TableName

-> Select * From TestTable
```

Show me every column in this table

```
Select Top ### * from TableName

-> Select Top 8 * from TestTable
```

Show me every column in the top 8 rows

-> Example Statement

```
Select Distinct * From TableName

-> Select Distinct * From TestTable
```

Distinct will only return unique or different values

-> Example Statement

```
Select Column1
     ,Column2
     ,Column3
from TableName
```

```
-> Select
   PatientName
       ,Visits
from TestTable
```

Specify each column

```
Select Distinct
    Column1
      ,Column2
      ,Column3
from TableName
```

```
-> Select Distinct
     PatientName
         ,Visits
from TestTable
```

Specify each Distinct column

-> Example Statement

Where Clause Rules
1. Used after the Tablename has been specified in the From
2. Used to return values with a specific condition
3. Can also be used in Update and Delete statements, which will be shown next section

-> Example Statement

```
-> Select
   PatientName
  ,PatientState
 from TestTable
where PatientState = 'GA'
```

Section 5: Where Clause Operators

Symbol	Description	Example

And	Filters values based on two conditions	`Select * from TableName`
		`where Column1 = 'Condition'`
		`and Column2 = 'Condition'`

Not	Filter values if condition is not true	`Select * from TableName` `where not Column1 = 'Condition'`

Or	Filters values based on one of two conditions	`Select * from TableName`
		`where Column1 = 'Condition'`
		`or Column2 = 'Condition'`

Like	Search for a patten of values within a column (Uses wildcard symbols)	`Select * from TestTable` `where Column1 like 'Condition1'`
Between	Returns values between values in a range	`Select * from TableName` `where Column1 between 'Condition1' and 'Condition2'`
In	Specify multiple values within a column	`Select * from TestTable` `where Column1 in ('Value1','Value2','Value3')`

Section 5: Wildcard Symbols

Functions	Description	Example
[]	Finds any character within the brackets	b[eai]d will find bed, bad, bid b[ea]d will find bed, bad, but not bid
_	Finds any single character	b_d will find bad, bed, bid, bod, bud
%	Finds any pattern after or before the symbol	da% will find data, date, day, dance, etc.
-	Finds a range of characters	b[a-e]d will find bad, bbd, bcd, bdd, bed

These wildcard symbols are used in Like conditions

Section 5: Order by

Order By Rules
1. An Order by statement sorts the results in ascending or descending order
2. If you don’t specify whether to sort by ascending or descending, then ascending is the default
3. You need to specify which column by either using the column number or name (see video example)

```
-> Select Distinct
   PatientName
  ,PatientState
from TestTable
where Gender = 'M'
Order by PatientState DESC
--ASC
--Order by 2 DESC --ASC
```

-> Example Statement

Common Functions	Description
Min	Returns the smallest value in the specified column
Max	Returns the largest value in the specified column
Count	Returns the number of rows
Sum	Returns the total sum of a numeric column
Avg	Returns the average of a numeric column

```
Select Sum(Value)
from TableName
```

-> Example Statement

```
-> Select Count(PatientName)
   From TestTable
```

Section 5: Group by

Group By Rules
1. A Group by Statement is necessary in SSMS and Azure Data studio when using aggregate functions
2. Groups rows into summary rows.
3. When using a group by without aggregates it acts like a distinct. Although they may return similar returns, they are different.

```
-> Select
       PatientState
       ,Count(*)
   From TestTable
   Group by
   PatientState
```

-> Example Statement

A group by is necessary if you are aggregating with another column (see video example)

-> Example Statement

```
Select Column1
    , Sum(Value)
from TableName
    Group by
    Column1

    -> Select
    PatientState
,Count(PatientName)
    From TestTable
Group by PatientState
```

In the previous example our aggregate column was not named.

```
Select Column1 as 'AliasName'
     , Sum(Value) as 'AliasName'
from TableName
  Group by
  Column1

-> Select
PatientState as 'State'
,Count(PatientName) as 'CountofPatients'
From TestTable
Group by PatientState
```

-> Example Statement

```
Select
Concat(Column1, Column2) as ‘AliasName'
,Count(Value) as ‘AliasName'
From TableName
Group by Column1, Column2

-> Select
Concat(PatientState,' - ',Gender) as 'StateGender'
,Count(Distinct PatientName) as 'CountofPatients'
From TestTable
Group by PatientState,Gender
```

-> Example Statement

Section 5: More Where and Having clause operators

Symbol	Description	Where Example	Having Example
=	Equal to	`Select * from TableName where Column1 = 'Condition'`	`Select * from TableName Group by Column1 Having Sum(Column1) = 'Condition'`
>	Greater than	`Select * from TableName where Column1 > 'Condition'`	`Select * from TableName Group by Column1 Having Sum(Column1) > 'Condition'`
<	Less than	`Select * from TableName where Column1 < 'Condition'`	`Select * from TableName Group by Column1 Having Sum(Column1) < 'Condition'`
>=	Greater than or equal	`Select * from TableName where Column1 >= 'Condition'`	`Select * from TableName Group by Column1 Having Sum(Column1) >= 'Condition'`
<=	Less than or equal	`Select * from TableName where Column1 <= 'Condition'`	`Select * from TableName Group by Column1 Having Sum(Column1) <= 'Condition'`

<> or !=	Not Equal	`Select * from TableName where Column1 <> 'Condition'`
		
		`Select * from TableName where Column1 != 'Condition'`
			`Select * from TableName Group by Column1 Having Sum(Column1) <> 'Condition'`
			
			`Select * from TableName Group by Column1 Having Sum(Column1) != 'Condition'`

The Having clause is very similar to the where clause. However, where cannot be used with aggregate functions.

Using a Group by is necessary when using the Having clause.

```
Select
Column1, Column2
From TableName
Group by Column1, Column2
Having Condition
```

```
-> Select distinct
     PatientName
    ,PatientState
From TestTable
Group by
     PatientName
    ,PatientState
Having Sum(Charges) between '900' and '10000'
```

-> Example Statement

Section 5: Having Example

```
-> Select top 3
    PatientName
   ,PatientState
From TestTable
Group by
PatientName
   ,PatientState
Having Sum(Charges) between
'900' and '10000'
order by 1 desc
```

VS.

```
-> Select top 3
    PatientName
   ,PatientState
from TestTable
where Charges between '900'
and '10000'
order by 1 desc
```

-> Example Statement

Case statement searches through specified columns finding instances where condition is met and returning specified values. Logic is if then

```
    Select
      Column
     ,Case When Column = 'Condition' Then 'Value'
           Else Column
            END
     From TestTable

-> Select
     PatientID
     ,PatientName
     ,Case When PatientState = 'GA' Then 'Georgia’
           Else PatientState
            END as 'PatientState’
   From TestTable
```

-> Example Statement

```
-> Select Distinct
    PatientID
    ,PatientName
    ,Case When PatientState = 'GA' Then 'Georgia'
  When PatientState = 'AL' Then 'Alabama'
  When PatientState = 'AK' Then 'Alaska'
  When PatientState = 'AZ' Then 'Arizona'
  When PatientState = 'UT' Then 'Utah'
  Else PatientState
          END as 'PatientState'
From TestTable
Where PatientState in ('GA','AL','AK','AZ','UT')
```

-> Example Statement

Section 5: Section Introduction

```sql
Select
PatientName
,PatientState
,Sum(Charges) as Charges
,Sum(Visits) as Visits
,Sum(Charges)/Sum(Visits) as AvgChargePerVisit
From TestTable
Where Gender = 'F'
and PatientState in ('GA','FL')
Group by
PatientName
,PatientState
Having Sum(Charges) >= 900
Order by PatientName
```

```sql
Select
From
Where
Group by
Having
Order by
```

Section 5: Self Evaluation

Step 1: Connect to SQLCourse_DB (or create this database if you haven't already
Step 2: Select PatientState, Gender, and average number of visits
Step 3: Write a case statement and spell out each gender
Step 4: Add a second case statement, when the average visits is greater than 10 then insert this value – "Greater than 10 visits"
Step 5: Sort by PatientState in ascending order
Step 6: Filter to only states with charges between 1,200 and 20,000
Step 7: Give each column an alias name

```
-> Example Statement
```

Section 6: Jumping Back to Inserting Data (section 4)

- *Updating Data*
- *Deleting Data*
- *Truncate Tables*
- *Alter Table*
- *Select Into*
  - *With conditions*
- *Temporary Tables*

Note be careful when updating data. Overwriting data cannot be reversed.

```
Update TableName
Set Column1 = value
Where condition
```

```
-> Update TestTable
Set PatientState = 'CA'
Where PatientID = '12354'
```

-> Example Statement

Section 6: Updating (Swapping) Values

***Note* be careful when updating data. Overwriting data cannot be reversed.**

```
Update TableName
Set Column1 = Column2,
Column2 = Column1
```

-> Example Statement

```
-> Update TestTable
Set PatientState = Gender,
Gender = PatientState
```

Section 6: Deleting Data

***Note* be careful when deleting data. Once data is deleted it cannot be reversed.**

```
Delete From TableName
Where condition
```

-> Example Statement

```
-> Delete From TestTable
Where PatientID = '12354'
```

Section 6: Truncating Tables

Truncating Tables is like deleting data and dropping a table. However, you delete all the data without dropping the table.

```
Truncate Table TableName

-> Truncate Table TestTable
```

-> Example Statement

Three ways you can alter a table

- Alter a column
- Add a column
- Drop a column

```
Alter Table TableName
Alter Column ColumnName datatype

Alter Table TableName
Add ColumnName datatype

Alter Table TableName
Drop Column ColumnName
```

-> Example Statement

Section 6: Alter Table Examples

```
-> Alter Table TestTable
   Alter Column Visits float

-> Alter Table TestTable
   Add PatientAge int
```

```
-> Update TestTable
   Set PatientAge = '25’
   Where PatientID = '12345'

-> Alter Table TestTable
   Drop Column PatientAge
```

-> Example Statement

```
-> Select
   Into
   From
   Where
   Group by
   Having
   Order by
```

```
-> Select
    PatientID
   ,PatientState
   ,Gender
   ,Visits
   ,Charges
   INTO TestTable2
   From TestTable
   Where Charges > 10000
```

-> Example Statement

```
-> Select
   Into
   From
   Where
   Group by
   Having
   Order by
```

```
-> Create Database Backup_SQLCourse_DB

   Select *
   INTO Backup_SQLCourse_DB.dbo.TestTable
   From TestTable
```

-> Example Statement

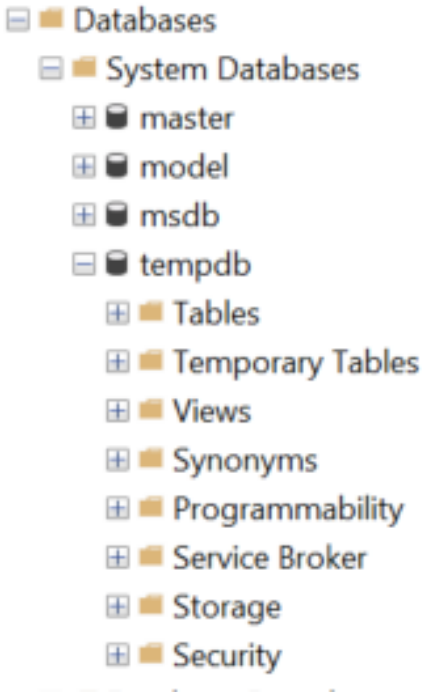
```
-> Select
   Into
   From
   Where
   Group by
   Having
   Order by
```

-> Example Statement

```
-> Select
Case When Gender = 'm' then 'Male'
 When Gender = 'f' then 'Female'
 Else null End as 'Gender'
,Sum(Visits) as Visits
,Sum(charges) as Charges
,Count(PatientID) as Number_of_Patients
Into TestTable3
From TestTable
Where PatientState in
('GA','FL','WY','UT','CA')
Group by Gender
Having Sum(Charges) > 10000
```

Section 6: Temporary Tables

Temporary Tables are useful when working with a very large datasets. You can create a subset of the tables that are needed to improve run times.




```
->Select
Case When Gender = 'm' then 'Male'
 When Gender = 'f' then 'Female'
 Else null End as 'Gender'
,Sum(Visits) as Visits
,Sum(charges) as Charges
,Count(PatientID) as Number_of_Patients
Into #TestTable
From TestTable
Where PatientState in
('GA','FL','WY','UT','CA')
Group by Gender
Having Sum(Charges) > 10000
```

-> Example Statement

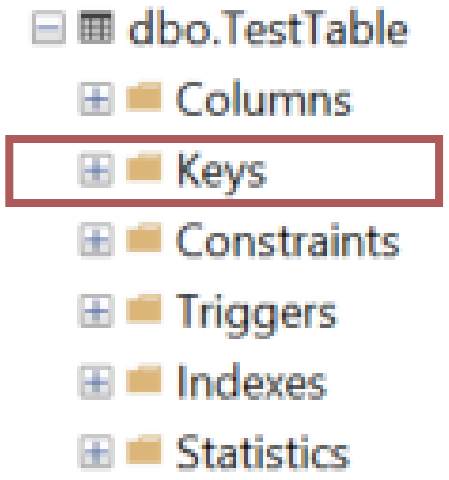
Section 6: Self Evaluation

Step 1: Connect to SQLCourse_DB (or create this database if you haven't already)
Step 2: Upate all of the values in the Gender Column to 'Male' and 'Female'
Step 3: Upate the Patients name to 'Bobby' where PatientID = '12348'
Step 4: Delete patients from 'FL'
Step 5: Alter the PatientState datatype to varchar(50)
Step 6: Put all the PatientNames and PatientIDs into
another table called 'PatientTable'
Step 7: Add a few columns to 'PatientTable' - Weight, Height, Age
*You decide the datatype

Section 7: Joining Tables

- *Primary Key Review*
- *Relational Databases*
- *Data Relationships*
- *Foreign Keys*
- *Setting up Primary and Foreign Keys*
- *Inner Join*
- *Left Joins*
- *Right Joins*
- *Review Cheat Sheet*

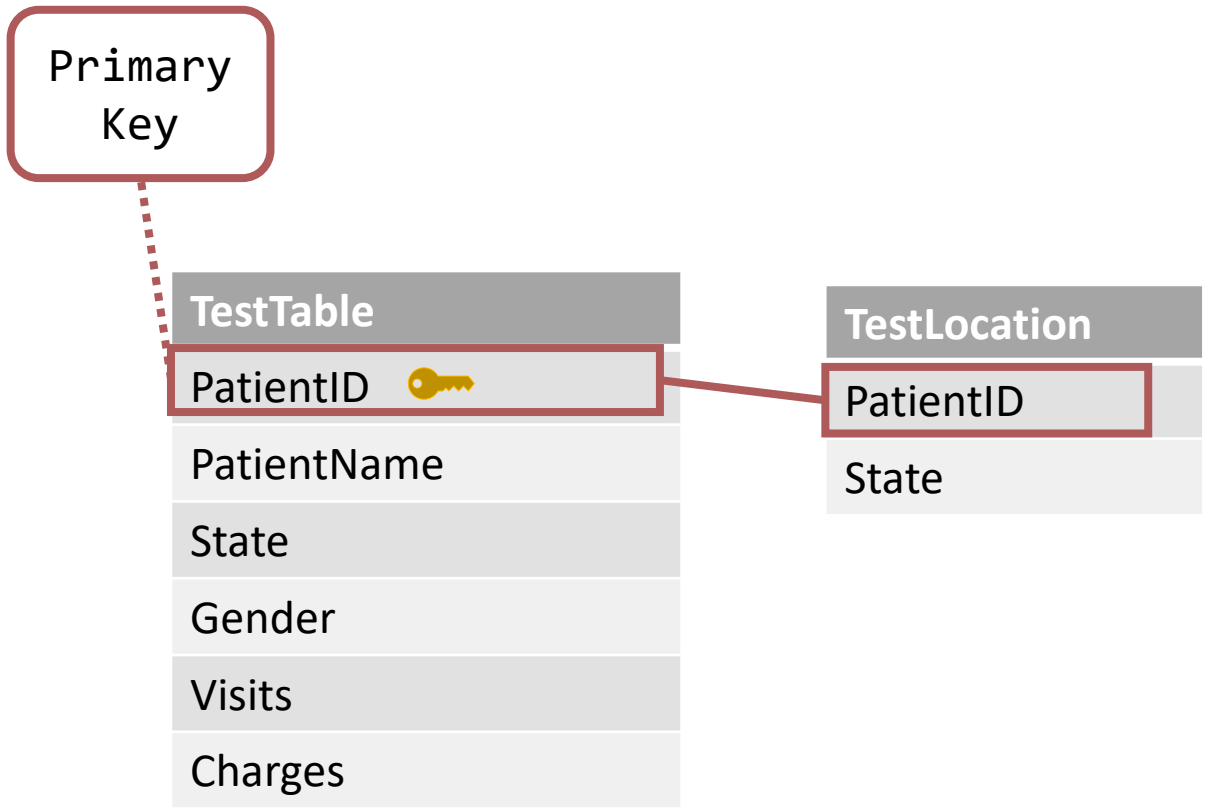
Database example



PatientID	PatientName	PatientSta	Gender	Visits	Charges
12355	Fred	CA	M	3	500
12355	Fred	CA	M	3	500
12355	Fred	CA	M	3	500
12355	Fred	CA	M	3	500
12355	Fred	CA	M	3	500
12355	Fred	CA	M	3	500

-> Example Statement

Section 7: Primary Key Review



PatientID	PatientName	Gender	Visits	Charges
12345	John	M	3	$ 200
12346	Jane	F	1	$ 400
12347	Alex	F	6	$ 900
12348	Bob	M	7	$ 8,000
12349	Josh	M	12	$ 19,000
12350	Stephanie	F	18	$ 25,000
12351	Amber	F	4	$ 400
12352	Brittany	F	6	$ 4,000
12353	Bill	M	8	$ 5,000
12354	Nate	M	22	$ 28,000

PatientID	State
12345	AL
12346	AK
12347	AZ
12348	CA
12349	CO
12350	FL
12351	GA
12352	GA
12353	UT
12354	WY

-> Example Statement

PatientID	PatientName	State	Gender	Visits	Charges
12345	John	AL	M	3	$ 200

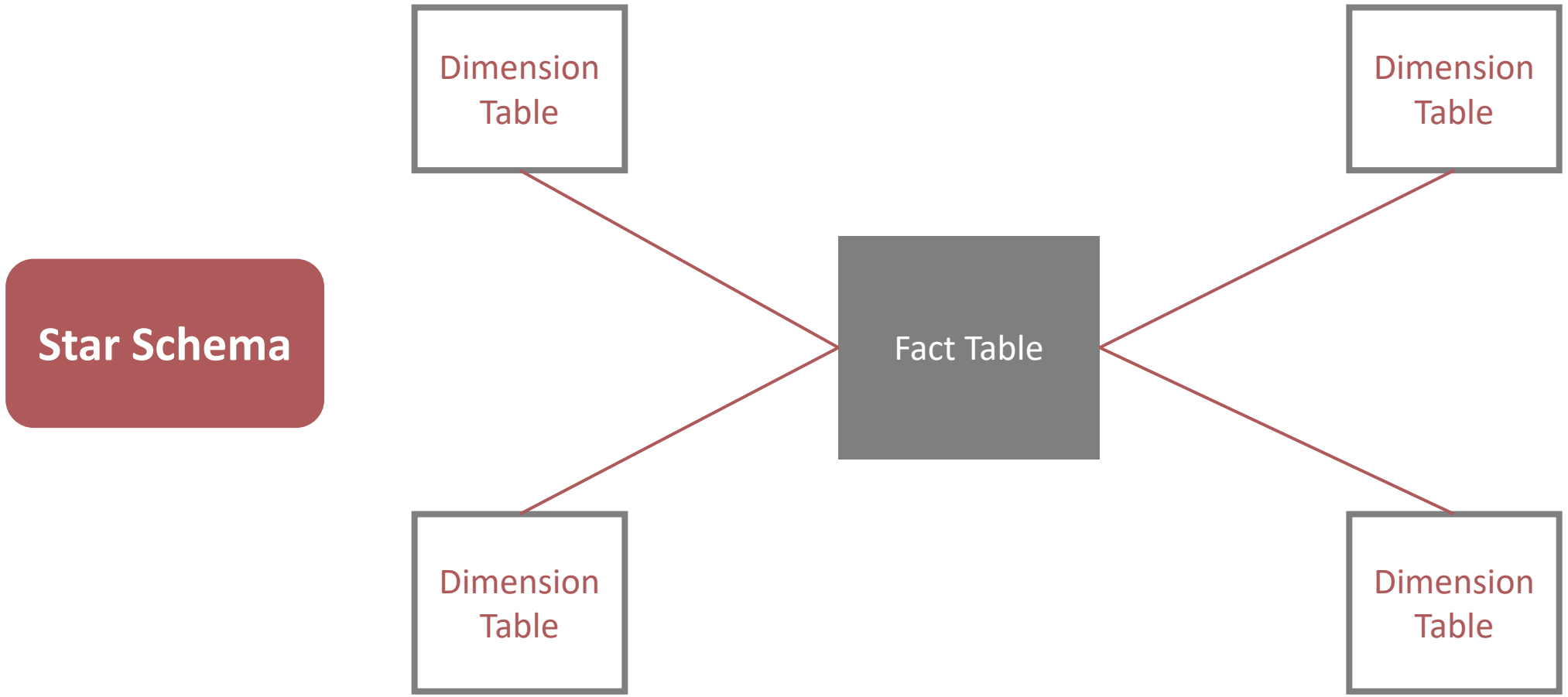
Section 7: Primary Key Review

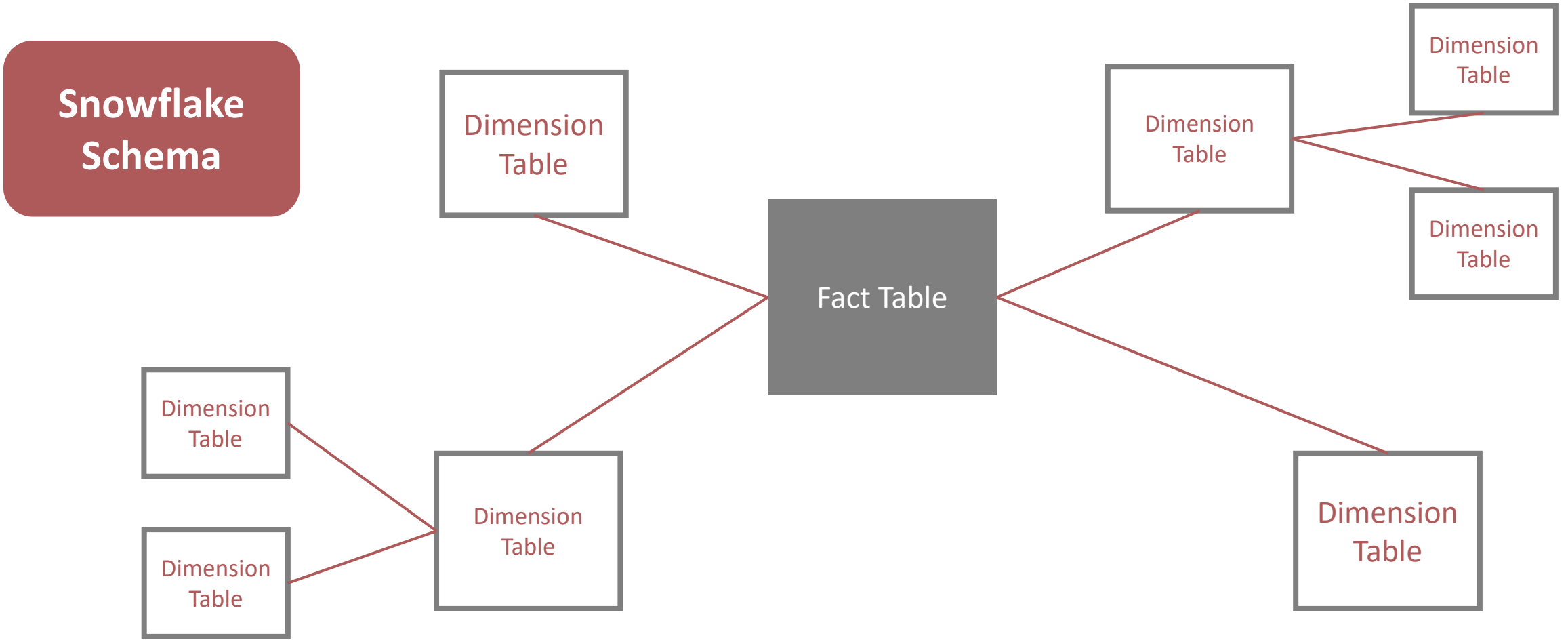
Primary Key Reminders

1. A Primary key must contain unique values
2. A table can only have one primary key
3. The primary key identifies each record in a table and can connect multiple tables together

-> Example Statement

```
-> CREATE TABLE TestTable
(PatientID int NOT NULL PRIMARY KEY,
PatientName varchar(255) NULL,
PatientState varchar(255) NULL,
Gender varchar(255) NULL,
Visits int NULL,
Charges int NULL Default 0)
```


Star Schema
Dimension Table
Dimension Table
Fact Table
Dimension Table
Dimension Table


Snowflake Schema
Dimension Table
Dimension Table
Dimension Table
Dimension Table
Fact Table
Dimension Table
Dimension Table
Dimension Table
Dimension Table

Why can’t we just store all the data in a single table?

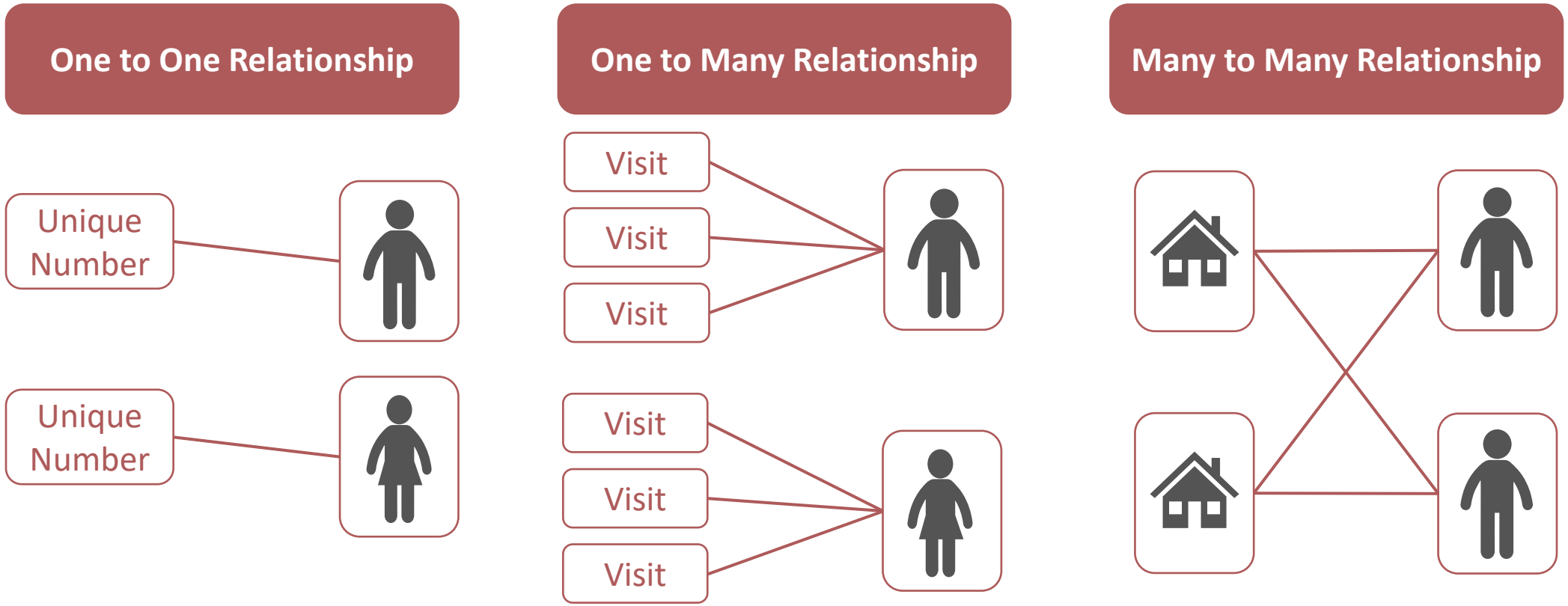
Section 7: Relational Databases

Patient Number	Visit ID	Patient Name	CPTCode	Date of Service	Location
12345	6789	Fred	99222	Jan. 1	West Clinic
12345	6789	Fred	90674	Jan. 1	West Clinic
12345	6790	Fred	99222	Jan. 5	West Hospital
12345	6791	Fred	99222	Jan. 6	East Hospital
12345	6791	Fred	96360 (IV)	Jan. 6	East Hospital
12345	6791	Fred	99222	Jan. 6	East Hospital

VS.

Patient Number	Visit ID	Date of Service	Location
12345	6789	Jan. 1	West Hospital
12345	6790	Jan. 5	West Clinic
12345	6791	Jan. 6	East Hospital

Visit ID	CPTCodes
6789	99222
6789	90674
6790	99222
6791	99222
6791	96360
6791	99222


One to One Relationship
Unique Number
Unique Number
One to Many Relationship
Visit
Visit
Visit
Visit
Visit
Visit
Many to Many Relationship

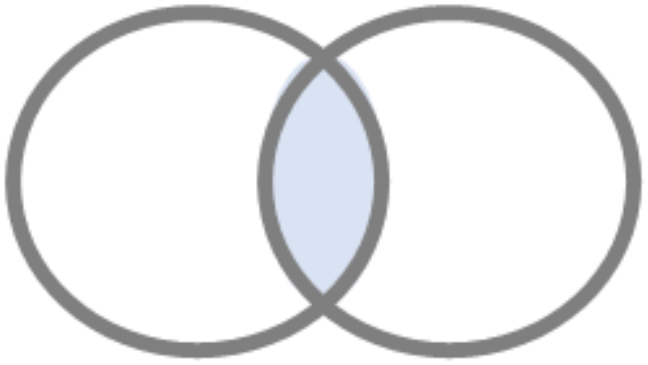


Table 1 Table 2

This is “left table”

```
Select *
   from TableName1
   INNER JOIN TableName2 on
     TableName1.Column = TableName2.Column


-> Select *
   from TestTable
   INNER JOIN HospitalTable on
     TestTable.LocationID = HospitalTable.LocationID
```

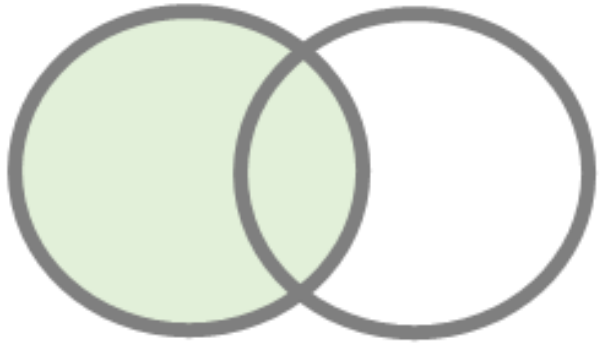


Table 1 Table 2

This is “left table”

```
Select *
   from TableName1
   LEFT JOIN TableName2 on
     TableName1.Column = TableName2.Column


-> Select *
   from TestTable
   LEFT JOIN HospitalTable on
     TestTable.LocationID = HospitalTable.LocationID
```

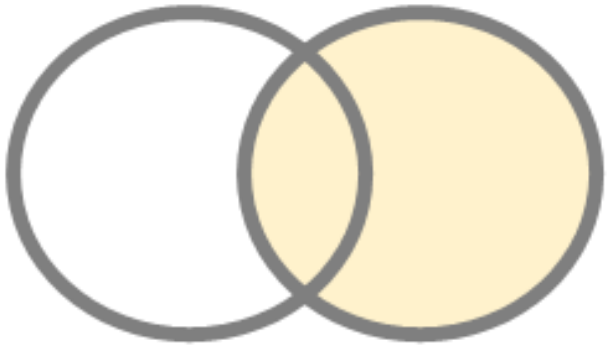


Table 1 Table 2

This is “left table”

```
Select *
   from TableName1
   RIGTH JOIN TableName2 on
     TableName1.Column = TableName2.Column


-> Select *
   from TestTable
   RIGHT JOIN HospitalTable on
     TestTable.LocationID = HospitalTable.LocationID
```

Section 7: Review Cheat Sheet

Basic Syntax

```
SELECT * FROM table_name
  -> Populates the whole table
SELECT column1, col2, col3… FROM table_name
  -> Populates specified columns
WHERE col2 = condition (=,>,<,>=,<=)
  -> filter rows where column values meet
condition
GROUP BY col1, col3
  -> Groups rows that have the same values
HAVING Count(*) > value
  -> Limit Aggregated Data
ORDER BY col4 (DESC or ASC)
  -> Order you results by a column
```

Useful Keywords when using SELECTS

```
DISTINCT -> Returns unique rows
BETWEEN a AND b -> Limits range of values
LIKE -> Pattern Search within the column values
IN(a,b,c) -> Returns values contained among list
TOP 100 -> Select top number of rows
```

Aggregation Functions

```
COUNT -> Count of rows SUM -> Cumulates values
AVG -> Avg's Values Max/Min -> Small/large values
```

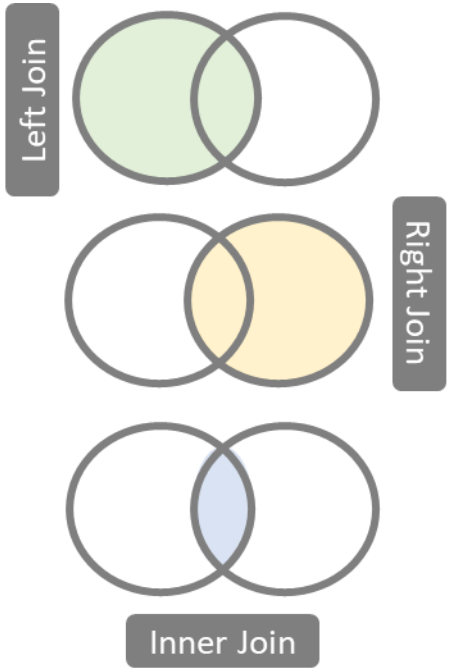
Table Manipulation

```
CREATE TABLE table_name (col1 datatype, col2 datatype…)
  -> Creates new table, specify the type of data in columns
DROP TABLE table_name
  -> Permanently deletes data table
TRUNCATE TABLE table_name
  -> Deletes data values in table, but table still exists
INSERT INTO table_name (col1, col2) VALUES (value1, value2)
  -> Insert data into created table
ALTER TABLE table_name ADD column_name datatype
  -> Add or Delete columns from table
UPDATE TABLE table_name SET col1 = value1, col2 = value2…
  -> Update existing records in a table
```

Joins

```
SELECT * FROM table1_name INNER JOIN table2_name ON
     table2_name.column1 = table1_name.column1
  -> Joining two tables using like columns
INNER JOIN -> Combing rows from tables where JOIN is true
LEFT JOIN -> Returns all records from left table and
matched records from the right table
RIGHT JOIN -> Returns all records from right table and
matched records from the left table
```

Joins Visualized



Section 8: Other Functions

- *Round*
- *Lower and Upper*
- *Isnull*
- *Inner Queries*
- *Stored Procedures*
- *Updating Data using another table*
- *Constraints*
- *Cast*
- *Convert*
- *Format*
- *Partition*
- *Left and Right Trim*

```
Select
Column1
,Column2
,Column3
,round(Column,2)
from TableName
```

Specifies number of decimals

```
-> Select
PatientID
,PatientName
,Visits
,round(Charges,2)
from TestTable
```

-> Example Statement

```
Select
isnull(Column1,’New Value')
,Column2
,Column2
from TableName
```

```
-> Select PatientID
,isnull(PatientName,'Not Provided')
,PatientState
,isnull(Gender, 'Unknown')
From TestTable
```

-> Example Statement

How many states have more than 2 patients?

A subquery is when a query is being referenced rather than a table

```
-> Select Count(*) as 'Count'
   From (
         Select
         PatientState
        ,Count(*) as 'Count'
        From TestTable
        Group by PatientState, Gender
        Having Count(*) >=2) InnerQuery
```

Subquery needs have an alias

-> Example Statement

How many patients have more visits than the average number of visits?

The condition is referencing another query rather than a condition/value

```
-> Select Count(*)
   from TestTable
   where Visits >=
        (Select Avg(visits) from TestTable)
```

-> Example Statement

Have you noticed we have used Select * from TestTable many many times? Lets make it a stored procedure

```
Create Procedure TableName
AS
SQLStatement


Exec Results
```

```
-> Create Procedure Results
AS
Select * from TestTable


-> Exec Results
```

-> Example Statement

Section 8: Format

```
FORMAT(Value, format, culture)
```

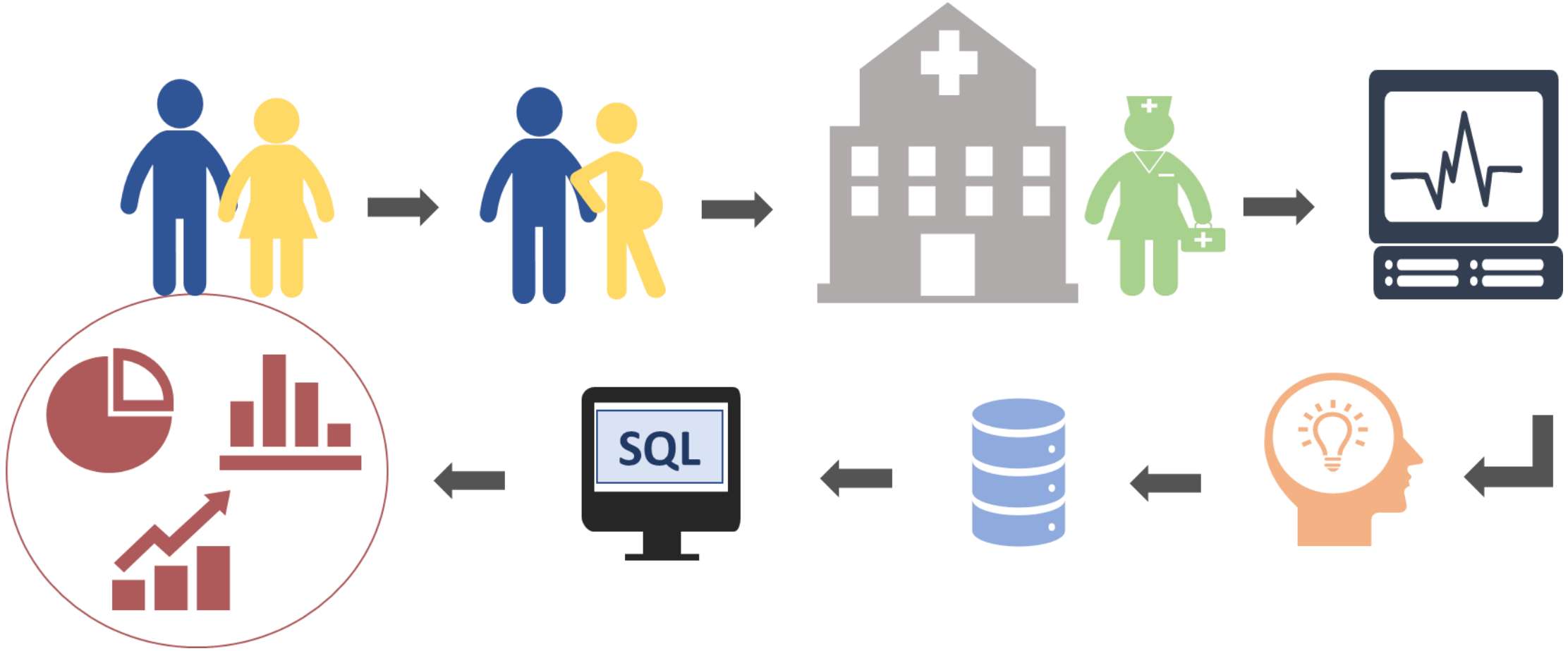
	Format
Percentage	P, P1, P2
Currency	C, C1, C2
Number	N or #
Date	D, D1, D2

	Culture
Percentage	P
Currency	C
Number	N or #
Date	D

Section 9: Real-world practice using healthcare data

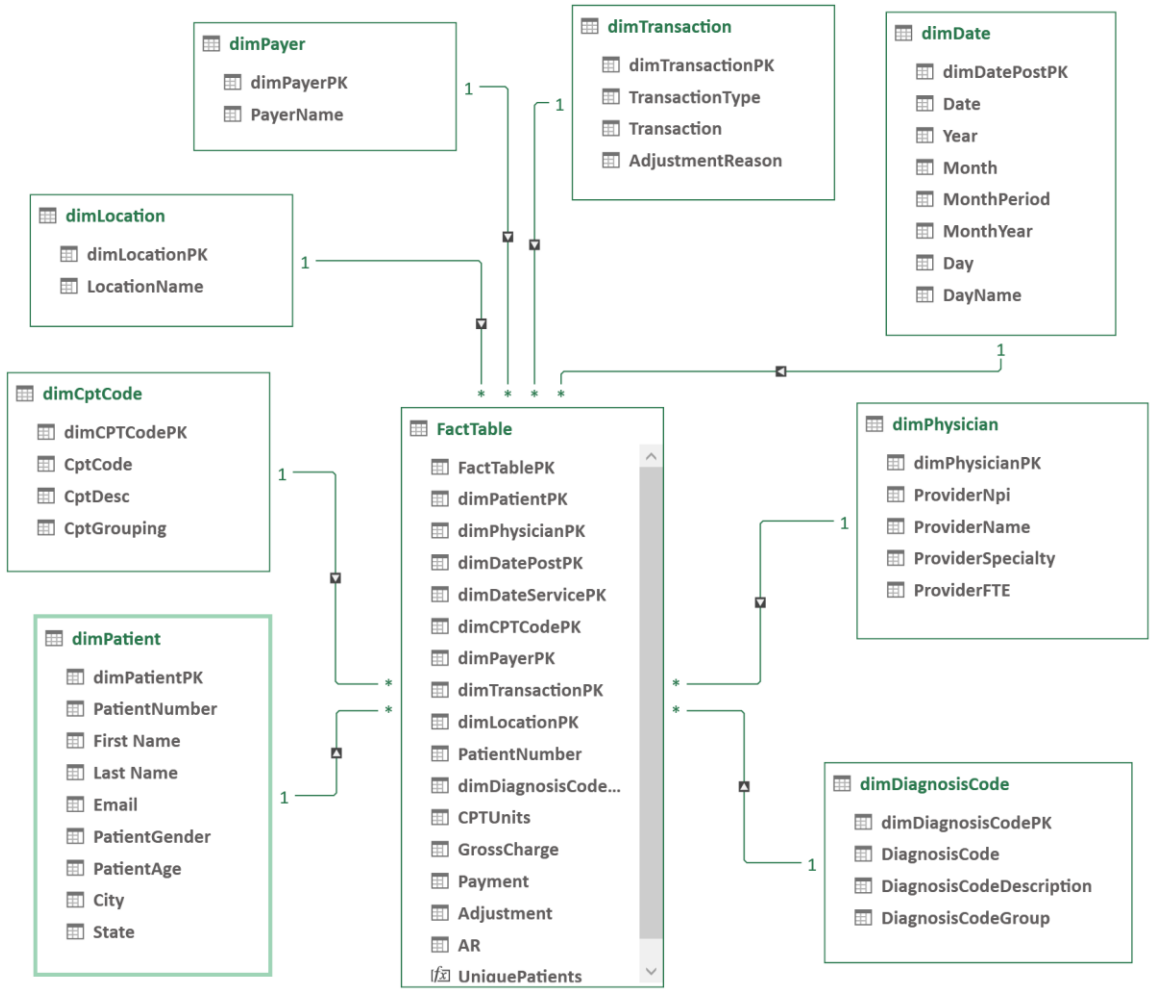
- *Review in Excel*
- *Review Entity Relationship Diagram*
- *Data Dictionary*
- *Upload data into SSMS*

Section 9: Example using Data



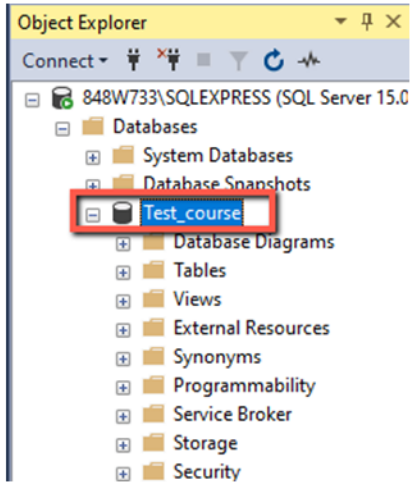
Section 9: Real-world healthcare data

Entity Relationship Diagram

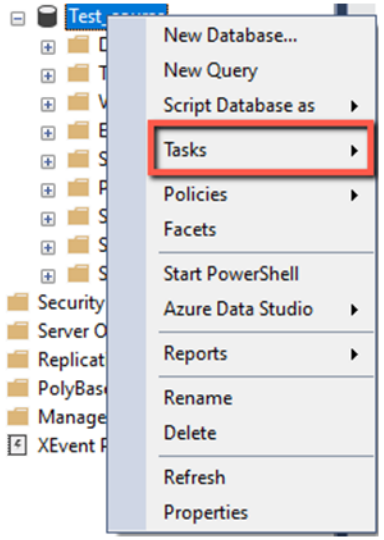


Section 9: Uploading Data in Microsoft SQL Server Management Studio

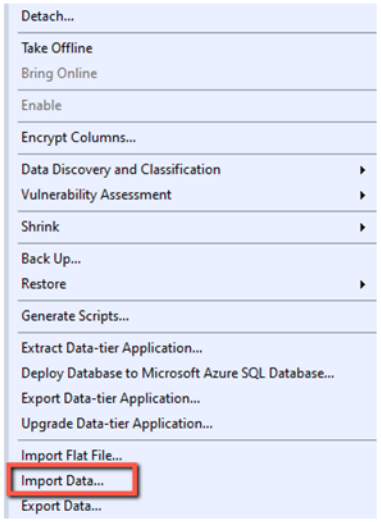
Step 1: Right click on the database you want to load excel file into.



Step 2: Click on "Tasks."

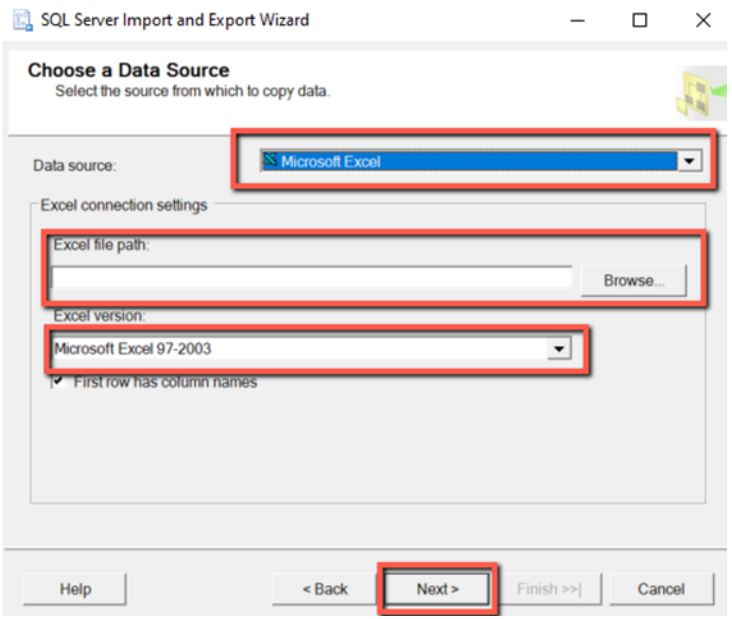


Step 3: Click "Import Data"



Step 4:This will open the import/export wizard, then Click "Next"

Step 5: Change the data source to "Microsoft Excel," the excel file using the "browse", then make sure you have the correct excel version.



Section 9: Uploading Data in Microsoft SQL Server Management Studio

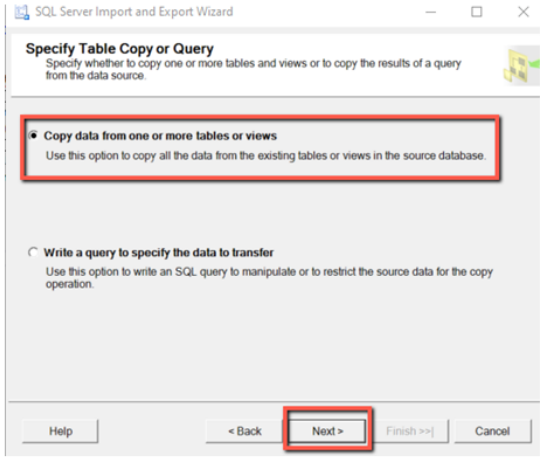
Step 6: It is possible that you will get an error here if you don't have updated import/export wizard software. If this is the case, then use the link below to update software then start at step1. If not, continue to step 7.

http://www.Microsoft.com/en-us/download/confirmation.aspx?id=13255

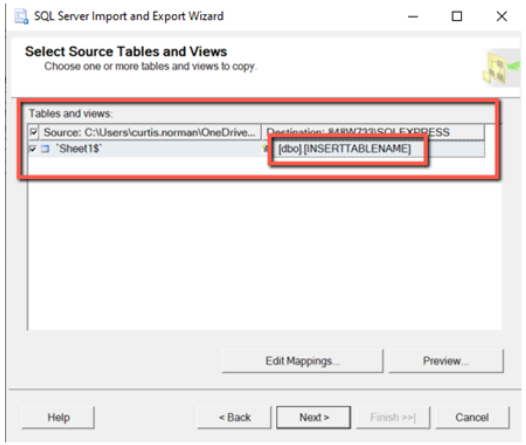
Step 7: Choose a destination: "SQL Server Native Client 11.0"

Step 8: Choose the server – my server is called "848W733\SQLExpress" (see image in step one to find your server's name). Then click "Next"

Step 9: Click "Copy Data from…" and "Next"



Step 10: Name the Table and click "Next"



Step 11: Click "Run Immediately" then click "Finish"

Step 12: The table successfully loaded when all the actions have a "success" status. Upon successfully load you can close the SQL server Import and Export Wizarad.

Section 10: Practice

- *Review in Excel*
- *Review Entity Relationship Diagram*
- *Data Dictionary*
- *Upload data into SSMS*

How many rows of data are in the FactTable that include a Gross Charge greater than $100?

How many unique patients exist is the Healthcare_DB?

How many CptCodes are in each CptGrouping?

How many physicians have submitted a Medicare insurance claim?

Calculate the Gross Collection Rate (GCR) for each LocationName

GCR = Payments divided GrossCharge
Which LocationName has the highest GCR?

How many CptCodes have more than 100 units?

Find the physician specialty that has received the highest amount of payments. Then show the payments by month for this group of physicians.

How many CptUnits by DiagnosisCodeGroup are assigned to a "J code" Diagnosis (these are diagnosis codes with the letter J in the code)?

You've been asked to put together a report that details Patient demographics. The report should group patients into three buckets- Under 18, between 18-65, & over 65
Please include the following columns:

-First and Last name in the same column
-Email
-Patient Age
-City and State in the same column

How many dollars have been written off (adjustments) due to credentialing (AdjustmentReason)? Which location has the highest number of credentialing adjustments? How many physicians at this location have been impacted by credentialing adjustments? What does this mean?

What is the average patientage by gender for patients seen at Big Heart Community Hospital with a Diagnosis that included Type 2 diabetes? And how many Patients are included in that average?

There are a two visit types that you have been asked to compare (use CptDesc).

- Office/outpatient visit est
- Office/outpatient visit new

Show each CptCode, CptDesc and the assocaited CptUnits.
What is the Charge per CptUnit? (Reduce to two decimals)
What does this mean?

Similar to Question 12, you've been asked to analysis the PaymentperUnit (NOT ChargeperUnit). You've been tasked with finding the PaymentperUnit by PayerName.

Do this analysis on the following visit type (CptDesc)

- Initial hospital care

Show each CptCode, CptDesc and associated CptUnits. What does this mean?

**Note you will encounter a zero value error. If you can't remember what to do find the ifnull lecture in Section 8.

Within the FactTable we are able to see GrossCharges. You've been asked to find the NetCharge, which means Contractual adjustments need to be subtracted from the GrossCharge (GrossCharges - Contractual Adjustments). After you've found the NetCharge then calculate the Net Collection Rate (Payments/NetCharge) for each physician specialty. Which physician specialty has the worst Net Collection Rate with a NetCharge greater than $25,000?
What is happening here? Where are the other dollars and why aren't they being collected?
What does this mean?

Build a Table that includes the following elements:

- LocationName
- CountofPhysicians
- CountofPatients
- GrossCharge
- AverageChargeperPatients