A Gentle Introduction to SQL

ICOS Big Data Summer Camp

May 10, 2016

Teddy DeWitt

(original slides from Mike Cafarella)

Learning Overview

- Why is SQL cool?
- Intro to schema and tables
- Running queries
- On-ramp for SQL – read MOAR books!

Relational Databases (1)

- A database is an organized collection of data
- A common kind is a *relational database*
- The software is called a Relational Database Management System (RDBMS)
  - Oracle, PostgreSQL, Microsoft's SQLServer, MySQL, SQLite, etc
- Your dataset is "a database", managed by an RDBMS

AID	**Name**	**Country**	**Sport**
1	Mary Lou Retton	USA	Gymnastics
2	Jackie Joyner-Kersee	USA	Track
3	Michael Phelps	USA	Swimming

Relational Databases (2)

- A relational database is a set of “relations” (aka tables)
- Each relation has two parts:
  - Instance (a data table, with rows (aka tuples, records), and columns (aka fields, attributes))
    - # Rows = cardinality
    - # Columns = degree
  - Schema
    - Relation name
    - Name and type for each column
    - E.g., Student (sid int, name varchar(128))
- Excel comparison?
  - Instances or Tables are like tabs
  - Schema is column headers and format cells (e.g., number, date, text)

Instance of Athlete Relation

AID	Name	Country	Sport
1	Mary Lou Retton	USA	Gymnastics
2	Jackie Joyner-Kersee	USA	Track
3	Michael Phelps	USA	Swimming

What is the schema? **(aid: integer, name: string, country: string, sport:string)**

Cardinality & Degree? **Cardinality = 3, Degree = 4**

Relational Query Languages

- An RDBMS does lots of things, but mainly:
  - Keeps data safe
  - Gives you a powerful query language
- RDBMS is responsible for efficient evaluation
  - System can optimize for efficient query execution, and still ensure that the answer does not change
- Most popular query language is SQL

Let’s make this table - Athlete

AID	Name	Country	Sport
1	Mary Lou Retton	USA	Gymnastics
2	Jackie Joyner-Kersee	USA	Track
3	Michael Phelps	USA	Swimming

Creating Relations in SQL

- Create the Athlete relation (table)

```
CREATE TABLE Athlete
(aid INTEGER,
 name CHAR(30),
 country CHAR(20),
 sport CHAR(20))
```

AID	Name	Country	Sport

Adding & Deleting Rows in SQL

```
INSERT INTO  Athlete (aid, name, country, sport)
VALUES  (1, ‘Mary Lou Retton’, ‘USA’, ‘Gymnastics’)
```

```
INSERT INTO  Athlete (aid, name, country, sport)
VALUES  (2, ‘Jackie Joyner-Kersee’, ‘USA’, ‘Track’)
```

```
INSERT INTO  Athlete (aid, name, country, sport)
VALUES  (3, ‘Michael Phelps’, ‘USA’, ‘Swimming’)
```

- And we are going to add another row!

```
INSERT INTO  Athlete (aid, name, country, sport)
VALUES  (4, ‘Johann Koss’, ‘Norway’, ‘Speedskating’)
```

Table. Athlete. Boom!

AID	Name	Country	Sport
1	Mary Lou Retton	USA	Gymnastics
2	Jackie Joyner-Kersee	USA	Track
3	Michael Phelps	USA	Swimming
4	Johann Koss	Norway	Speedskating

Getting Data in SQL (1)

- SELECT all of the rows and columns:

```
SELECT *
FROM Athlete
```

AID	Name	Country	Sport
1	Mary Lou Retton	USA	Gymnastics
2	Jackie Joyner-Kersee	USA	Track
3	Michael Phelps	USA	Swimming
4	Johann Koss	Norway	Speedskating

- Only names and sports:

```
SELECT name, sport
FROM Athlete
```

```
SELECT A.name, A.sport
FROM Athlete A
```

Name	Sport
Mary Lou Retton	Gymnastics
Jackie Joyner-Kersee	Track
Michael Phelps	Swimming
Johann Koss	Speedskating

Getting Data in SQL (2)

AID	Name	Country	Sport
1	Mary Lou Retton	USA	Gymnastics
2	Jackie Joyner-Kersee	USA	Track
3	Michael Phelps	USA	Swimming
4	Johann Koss	Norway	Speedskating

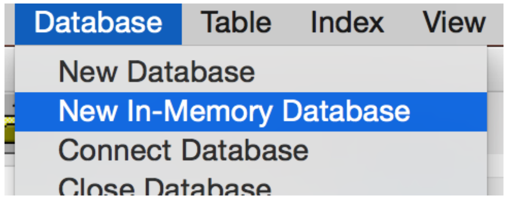
- SELECT names and sports WHERE country is USA:

```
SELECT A.name, A.sport
FROM Athlete A
WHERE A.country = ‘USA’
```

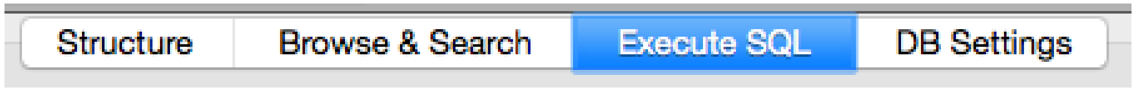
Name	Sport
Mary Lou Retton	Gymnastics
Jackie Joyner-Kersee	Track
Michael Phelps	Swimming

Hands-On #1

- Open Firefox SQLite Manager and select New In-Memory Database from the Database menu.



- Click "Execute SQL".



- In another window, go to web.eecs.umich.edu/~michjc/players.txt
- Copy the text into the "Enter SQL" box and click "Run SQL"

Hands-On #1

- Write queries to find:
  - Names of all the players in the database
  - All info for all players from Detroit
  - Names and teams of the first basemen (Position ID: 3)

Hands-On #1

- *Names of all the players in the database*

```
SELECT playerID FROM Allstars
```

- *All info for all players from Detroit*

```
SELECT *
FROM Allstars
WHERE teamID = "DET"
```

- *Names and teams of the first basemen*

```
SELECT playerID, teamID
FROM Allstars
WHERE startingPos = 3
```

Basic SQL Query

Attributes from input relations

Optional

List of relations

SELECT [DISTINCT] attr-list
FROM relation-list
WHERE qualification

Attr1 **op** Attr2
OPS: <, >, =, <=, >=, <>
Combine using AND, OR, NOT

(Conceptual) Evaluation:

1. Take cross-product of relation-list
2. Select rows satisfying qualification
3. Project columns in attr-list
   (eliminate duplicates only if DISTINCT)

Example of Basic Query(1)

- Schema:
  - Sailors (sid, sname, rating, age)
  - Boats (bid, bname, color)
  - Reserves (sid, bid, day)

Example of Basic Query(2)

Boats

bid	bname	color
101	jeff	red
103	boaty	black

Sailors

sid	sname	rating	age
22	dustin	7	45
58	rusty	10	35
31	lubber	8	55

Reserves

sid	bid	day
22	101	Oct-10
58	103	Nov-12
58	103	Dec-13

Example of Basic Query(3)

- Schema:
  - Sailors (sid, sname, rating, age)
  - Boats (bid, bname, color)
  - Reserves (sid, bid, day)
- Find the names of sailors who have reserved boat #103
- Are the names of the sailors and the numbers of the boats reserved in the same place?
- Must Join the tables

Example of Basic Query(4)

Reserves x Sailors

sid	bid	day	sid	sname	rating	age
22	101	Oct-10	22	dustin	7	45
22	101	Oct-10	58	rusty	10	35
22	101	Oct-10	31	lubber	8	55
58	103	Nov-12	22	dustin	7	45
58	103	Nov-12	58	rusty	10	35
58	103	Nov-12	31	lubber	8	55
58	103	Dec-13	22	dustin	7	45
58	103	Dec-13	58	rusty	10	35
58	103	Dec-13	31	lubber	8	55

Example of Basic Query(5)

- Find the names of sailors who have reserved boat #103

```
SELECT S.sname
FROM Sailors S, Reserves R
WHERE S.sid = R.sid AND R.bid = 103
```

sname
rusty
rusty

Using DISTINCT

3. Project columns in attr-list
   (eliminate duplicates only if DISTINCT)

```
SELECT DISTINCT sname
FROM Sailors S, Reserves R
WHERE S.sid = R.sid AND R.bid = 103
```

What's the effect of adding DISTINCT?

sname
rusty

Another Example

- Schema:
  - Sailors (<u>sid</u>, sname, rating, age)
  - Boats (<u>bid</u>, bname, color)
  - Reserves (<u>sid, bid, day</u>)
- Find the colors of boats reserved by a sailor named rusty

```
SELECT B.color
FROM Sailors S, Reserves R, Boats B
WHERE S.sid = R.sid AND R.bid = B.bid AND
   S.sname = 'rusty'
```

Hands-On #2

- SQLite Manager -> Database menu -> New In-Memory Database
- In another window, go to web.eecs.umich.edu/~michjc/teams.txt
- Copy the text, Run SQL, etc.
- In addition to Allstars table, Teams table:
  - yearID, lgID, teamID, franchID, name, park, attendance, BPF, PPF, teamIDBR, teamIDlahman45, teamIDretro

Hands-On #2

- Write queries to find:
  - Team names for all teams with attendance more than 2,000,000
  - Player ID and home stadium for all Allstars
  - TeamID, attendance for teams that had an all-star player

Hands-On #2

- *Team names for all teams with attendance more than 2,000,000*

```
SELECT name
FROM Teams
WHERE attendance > 2000000
```

- *Player ID and home stadium for all Allstars*

```
SELECT playerID, park
FROM Allstars A, Teams T
WHERE A.teamID = T.teamID
```

Hands-On #2

- *TeamID, attendance values for teams that had an all-star player*
- One answer:
  - SELECT A.teamID, attendance
    FROM Teams T, Allstars A
    WHERE T.teamID = A.teamID
- A better answer:
  - SELECT DISTINCT A.teamID, attendance
    FROM Teams T, Allstars A
    WHERE T.teamID = A.teamID

ORDER BY clause

- Most of the time, results are unordered
- You can sort them with the ORDER BY clause

Attribute(s) in ORDER BY clause must be in SELECT list.

Find the names and ages of all sailors, in increasing order of age

```
SELECT S.sname, S.age
FROM Sailors S
ORDER BY S.age [ASC]
```

Find the names and ages of all sailors, in decreasing order of age

```
SELECT S.sname, S.age
FROM Sailors S
ORDER BY S.age DESC
```

ORDER BY clause

```
SELECT S.sname, S.age, S.rating
FROM Sailors S
WHERE S.age > 20
ORDER BY S.age ASC, S.rating DESC
```

What does this query compute?

Find the names, ages, & ratings of sailors over the age of 20.

Sort the result in <u>increasing</u> order of age.

If there is a tie, sort those tuples in decreasing order of rating.

Hands-On #3

- Use the database loaded last time
- A twist:
  - Find TeamID and attendance values for teams that had an all-star player ORDERED BY ATTENDANCE

Hands-On #3

- *Find TeamID and attendance values for teams that had an all-star player ORDERED BY ATTENDANCE*

```
SELECT DISTINCT A.teamID, attendance
FROM Teams T, Allstars A
WHERE T.teamID = A.teamID
ORDER BY attendance DESC
```

Aggregate Operators

COUNT (*)
COUNT ([DISTINCT] A)
SUM ([DISTINCT] A)
AVG ([DISTINCT] A)
MAX (A) *Can use Distinct*
MIN (A) *Can use Distinct*

single column

```
SELECT  COUNT (*) FROM  Sailors S
```

```
SELECT COUNT (DISTINCT S.name)
FROM Sailors S
```

```
SELECT  AVG (S.age)
FROM  Sailors S
WHERE  S.rating=10
```

```
SELECT  AVG ( DISTINCT S.age)
FROM  Sailors S
WHERE  S.rating=10
```

Hands-On #4

- Use our previous Allstar and Teams DB
- Find:
  - Average attendance for all teams
  - Average attendance among teams that had an all-star player

Hands-On #4

- *Average attendance for all teams*

```
SELECT AVG(attendance)
FROM Teams
```

- *Average attendance among teams that had an all-star player*

```
SELECT AVG(DISTINCT attendance)
FROM Teams T, Allstars A
WHERE T.teamID = A.teamID
```

GROUP BY

- Conceptual evaluation
  - Partition data into groups according to some criterion
  - Evaluate the aggregate for each group

Example: *For each rating level, find the age of the youngest sailor*

```
SELECT  MIN (S.age), S.rating
FROM  Sailors S
GROUP BY  S.rating
```

How many tuples in the result?

Excel Equivalent: *Think about the results you would want from a pivot table….*

Hands-On #5

- With our same old database, first try a simple one:
  - Show all teamIds that had an all-star, along with number of all-star players

Hands-On #5

- *Show all teamIds that had an all-star, along with number of all-star players*

```
SELECT teamID, COUNT(*)
FROM Allstars
GROUP BY teamID
```

Hands-On #5

- Harder:
  - Show all team names that had an all-star, along with number of all-star players

Hands-On #5

- *Show all team names that had an all-star, along with number of all-star players*

```
SELECT name, COUNT(A.playerID)
FROM Allstars A, Teams T
WHERE A.teamID = T.teamID
GROUP BY T.name
```

Hands-On #5

- Even Harder:
  - Show all team names that had an all-star, along with number of all-star players, SORTED IN DESCENDING ORDER BY NUMBER OF ALL-STARS

Hands-On #5

- *Show all team names that had an all-star, along with number of all-star players, SORTED IN DESCENDING ORDER BY NUMBER OF ALL-STARS*

```
SELECT name, COUNT(A.playerID) AS playerCount
FROM Allstars A, Teams T
WHERE A.teamID = T.teamID
GROUP BY name
ORDER BY playerCount DESC
```

NULL Values in SQL

- NULL represents ‘unknown’ or ‘inapplicable’
- WHERE clause eliminates rows that don’t evaluate to true

What does this query return?

```
SELECT sname
FROM sailors
WHERE age > 45
        OR age <= 45
```

sailors

sid	sname	rating	age
22	dustin	7	45
58	rusty	10	NULL
31	lubber	8	55

Yes, it returns just dustin and lubber!

NULL Values in Aggregates

- NULL values generally ignored when computing aggregates

```
SELECT AVG(age)
FROM sailors
```

Returns 50!

sailors

sid	sname	rating	age
22	dustin	7	45
58	rusty	10	NULL
31	lubber	8	55

Questions?

Useful Resoruces

- URLS
  - http://www.w3schools.com/sql/
  - http://www.tutorialspoint.com/sqlite/
  - http://www.tutorialspoint.com/sqlite/sqlite_python.htm
- Books
  - Learning SQL – Alan Beaulieu
- Online Courses
  - Udemy – The Complete SQL Bootcamp ($)

SQL and Corporate Director Networks

ICOS Big Data Summer Camp

May 10, 2016

Teddy DeWitt

(original slides from Mike Cafarella)

Learning Overview

- Quick Review
- Joins
- Four questions
- Use SQL, get a dissertation
- Visualizing Networks

Basic SQL Query

SELECT [DISTINCT] attr-list

FROM relation-list

WHERE qualification

GROUP BY

ORDER BY

Attributes from input relations

List of relations

Attr1 **op** Attr2

OPS: <, >, =, <=, >=, <>

Combine using AND, OR, NOT

Partition Data into Groups

Sort data if you would like

The Power of Joins

- *Show all team names that had an all-star, along with number of all-star players, SORTED IN DESCENDING ORDER BY NUMBER OF ALL-STARS*

```
SELECT name, COUNT(A.playerID) AS playerCount
FROM Allstars A, Teams T
WHERE A.teamID = T.teamID
GROUP BY name
ORDER BY playerCount DESC
```

This is a JOIN.

The Power of Joins (2)

- *Show all team names that had an all-star, along with number of all-star players, SORTED IN DESCENDING ORDER BY NUMBER OF ALL-STARS*

```
SELECT name, COUNT(A.playerID) AS playerCount
FROM Allstars A
INNER JOIN Teams T
ON A.teamID = T.teamID
GROUP BY name
ORDER BY playerCount DESC
```

This too is a JOIN.

The Power of Joins (3)

- *There needs to be a common identifier between tables for the join to be useful*

- *Could you join a table with itself…..*

Board of Directors

- *What is a board of directors?*
- *What is a board interlock?*
- *What is a social network?*
- *What do I need to create a social network map of board interlocks?*

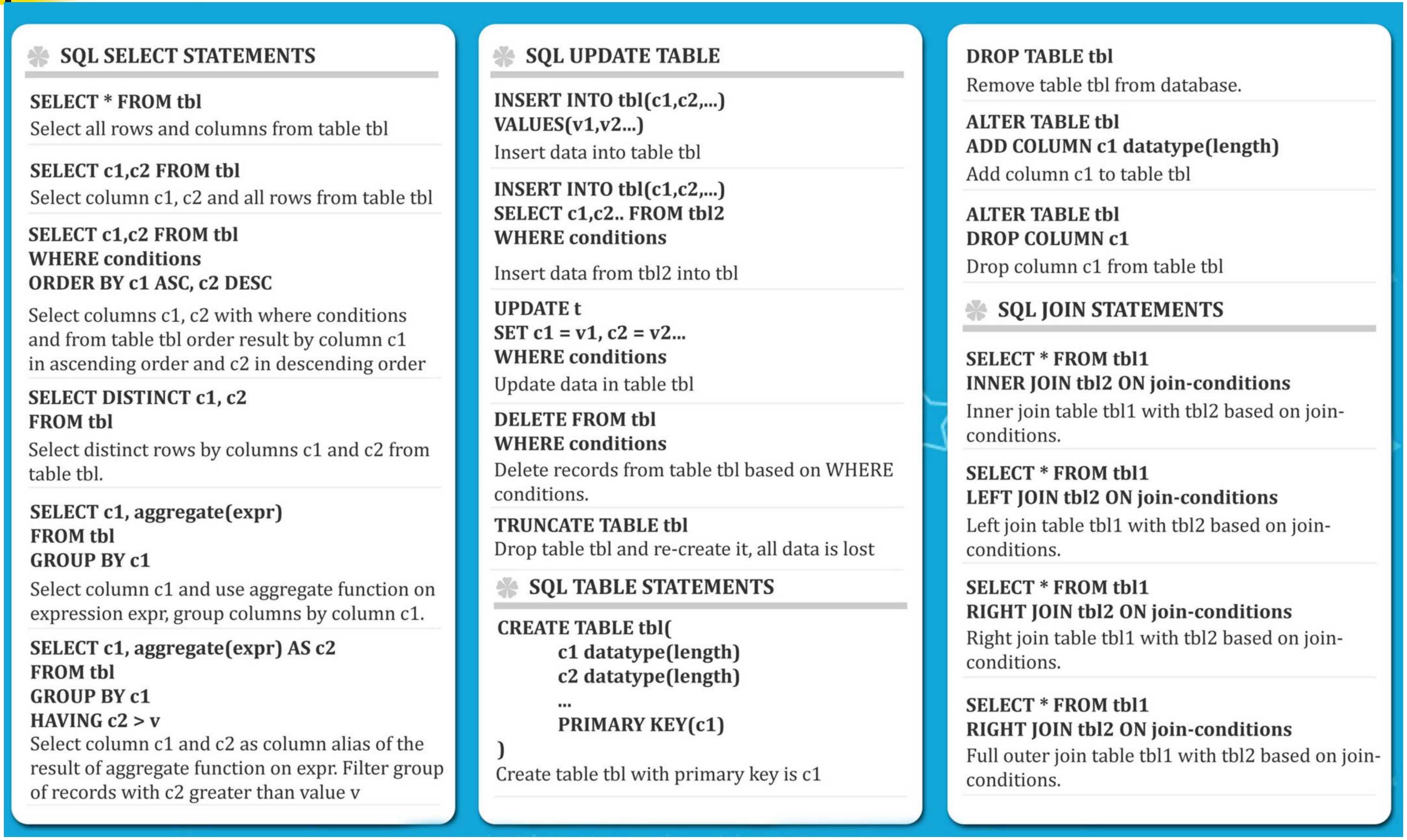
SQL and Networks

DEMO

SQL and Networks

Questions?

APPENDIX



SQL SELECT STATEMENTS

SELECT * FROM tbl
Select all rows and columns from table tbl

SELECT c1,c2 FROM tbl
Select column c1, c2 and all rows from table tbl

SELECT c1,c2 FROM tbl
WHERE conditions
ORDER BY c1 ASC, c2 DESC
Select columns c1, c2 with where conditions and from table tbl order result by column c1 in ascending order and c2 in descending order

SELECT DISTINCT c1, c2
FROM tbl
Select distinct rows by columns c1 and c2 from table tbl.

SELECT c1, aggregate(expr)
FROM tbl
GROUP BY c1
Select column c1 and use aggregate function on expression expr, group columns by column c1.

SELECT c1, aggregate(expr) AS c2
FROM tbl
GROUP BY c1
HAVING c2 > v
Select column c1 and c2 as column alias of the result of aggregate function on expr. Filter group of records with c2 greater than value v

SQL UPDATE TABLE

INSERT INTO tbl(c1,c2,...)
VALUES(v1,v2...)
Insert data into table tbl

INSERT INTO tbl(c1,c2,...)
SELECT c1,c2.. FROM tbl2
WHERE conditions
Insert data from tbl2 into tbl

UPDATE t
SET c1 = v1, c2 = v2...
WHERE conditions
Update data in table tbl

DELETE FROM tbl
WHERE conditions
Delete records from table tbl based on WHERE conditions.

TRUNCATE TABLE tbl
Drop table tbl and re-create it, all data is lost

SQL TABLE STATEMENTS

```
CREATE TABLE tbl(
        c1 datatype(length)
        c2 datatype(length)
        ...
        PRIMARY KEY(c1)
)
```
Create table tbl with primary key is c1

DROP TABLE tbl
Remove table tbl from database.

ALTER TABLE tbl
ADD COLUMN c1 datatype(length)
Add column c1 to table tbl

ALTER TABLE tbl
DROP COLUMN c1
Drop column c1 from table tbl

SQL JOIN STATEMENTS

SELECT * FROM tbl1
INNER JOIN tbl2 ON join-conditions
Inner join table tbl1 with tbl2 based on join-conditions.

SELECT * FROM tbl1
LEFT JOIN tbl2 ON join-conditions
Left join table tbl1 with tbl2 based on join-conditions.

SELECT * FROM tbl1
RIGHT JOIN tbl2 ON join-conditions
Right join table tbl1 with tbl2 based on join-conditions.

SELECT * FROM tbl1
RIGHT JOIN tbl2 ON join-conditions
Full outer join table tbl1 with tbl2 based on join-conditions.

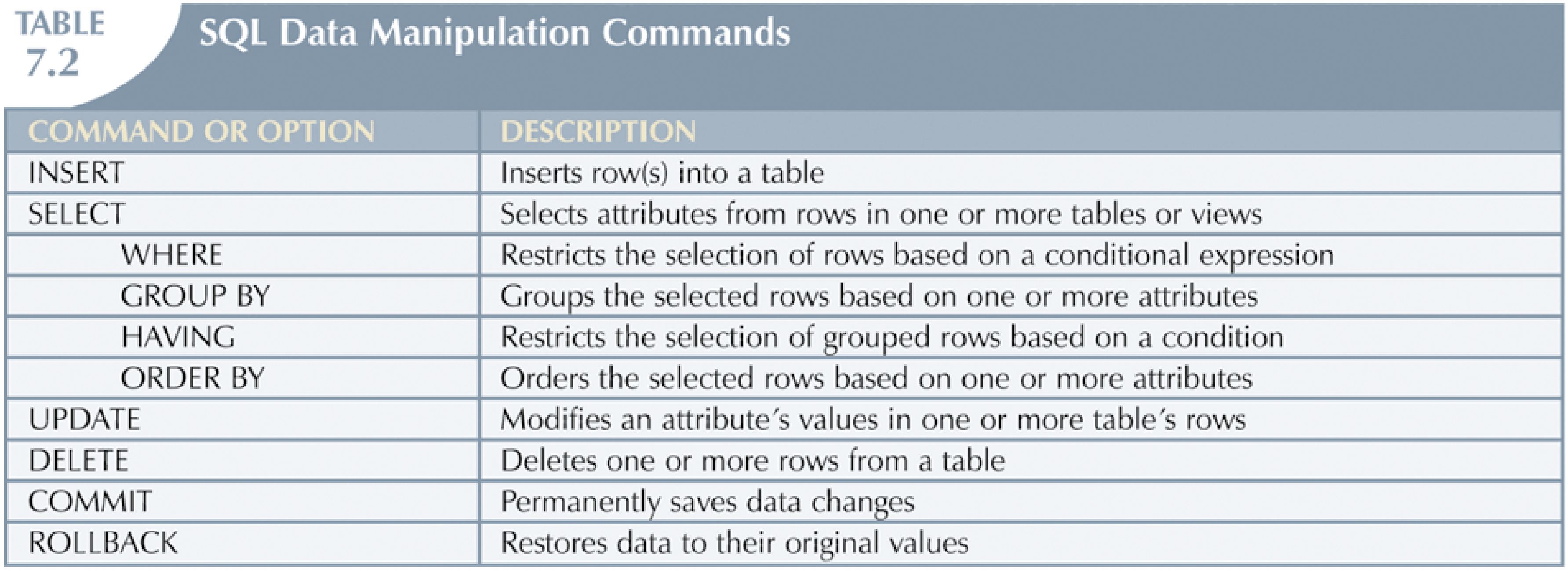


TABLE 7.2 SQL Data Manipulation Commands

COMMAND OR OPTION	DESCRIPTION
INSERT	Inserts row(s) into a table
SELECT	Selects attributes from rows in one or more tables or views
WHERE	Restricts the selection of rows based on a conditional expression
GROUP BY	Groups the selected rows based on one or more attributes
HAVING	Restricts the selection of grouped rows based on a condition
ORDER BY	Orders the selected rows based on one or more attributes
UPDATE	Modifies an attribute's values in one or more table's rows
DELETE	Deletes one or more rows from a table
COMMIT	Permanently saves data changes
ROLLBACK	Restores data to their original values

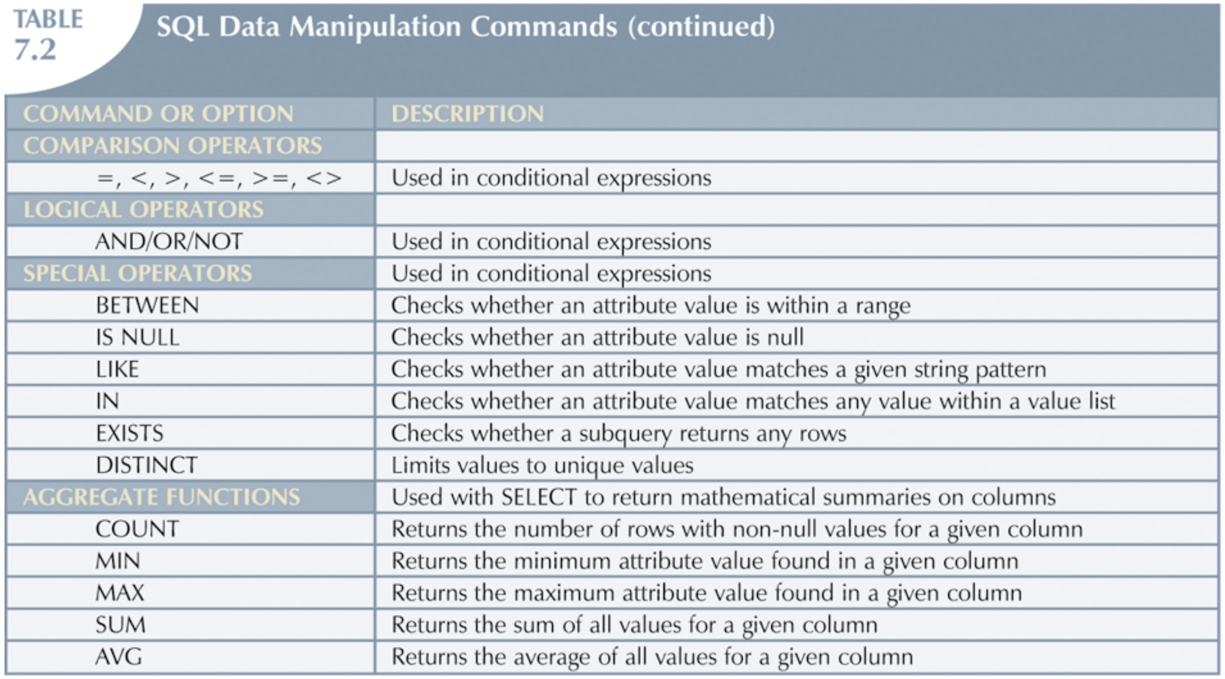


TABLE 7.2 SQL Data Manipulation Commands (continued)

COMMAND OR OPTION	DESCRIPTION
COMPARISON OPERATORS	
=, <, >, <=, >=, <>	Used in conditional expressions
LOGICAL OPERATORS	
AND/OR/NOT	Used in conditional expressions
SPECIAL OPERATORS	Used in conditional expressions
BETWEEN	Checks whether an attribute value is within a range
IS NULL	Checks whether an attribute value is null
LIKE	Checks whether an attribute value matches a given string pattern
IN	Checks whether an attribute value matches any value within a value list
EXISTS	Checks whether a subquery returns any rows
DISTINCT	Limits values to unique values
AGGREGATE FUNCTIONS	Used with SELECT to return mathematical summaries on columns
COUNT	Returns the number of rows with non-null values for a given column
MIN	Returns the minimum attribute value found in a given column
MAX	Returns the maximum attribute value found in a given column
SUM	Returns the sum of all values for a given column
AVG	Returns the average of all values for a given column

Useful Resoruces

- URLS
  - http://www.w3schools.com/sql/
  - http://www.tutorialspoint.com/sqlite/
  - http://www.tutorialspoint.com/sqlite/sqlite_python.htm
- Books
  - Learning SQL – Alan Beaulieu
- Online Courses
  - Udemy – The Complete SQL Bootcamp ($)