GENERAL ASSEMBLY

SQL BOOTCAMP PART II

Joana Wang

SQL BOOTCAMP

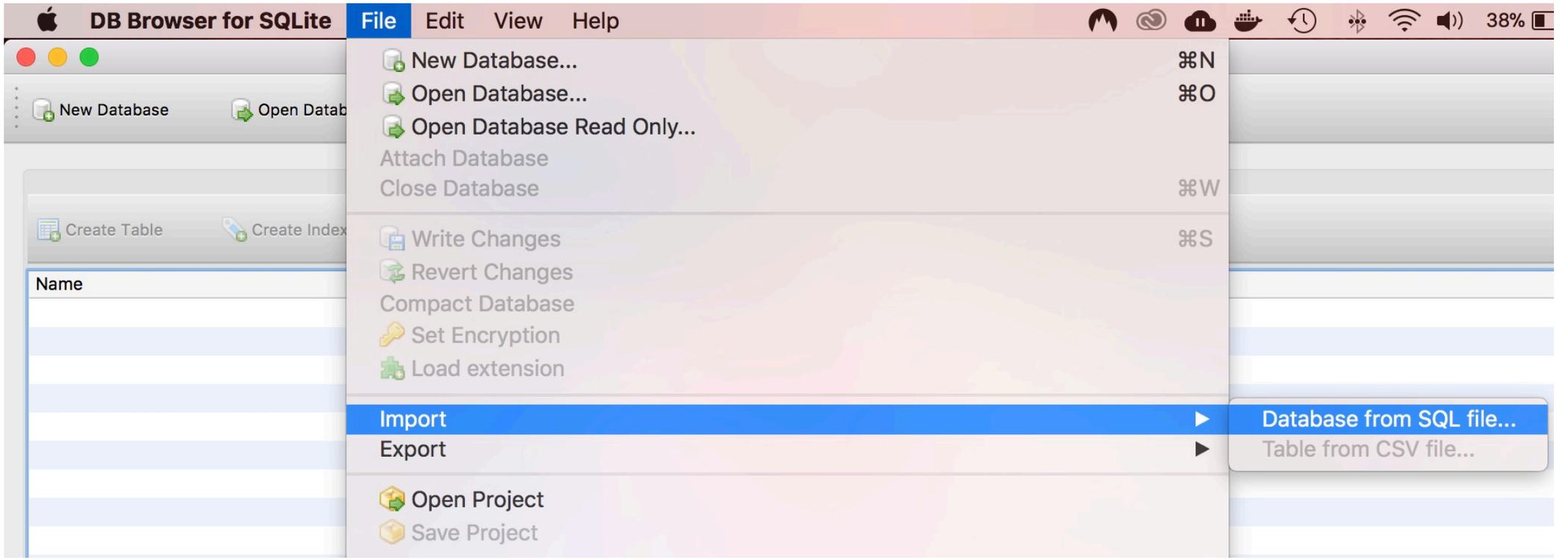
INTRODUCTION TO THE COURSE

INSTALLFEST!

- Save and unzip all files in a folder
- Database interface: http://sqlitebrowser.org/
- Text editor: https://www.sublimetext.com/

INSTALLFEST!

- Import data from file **iowa.db.sql**
- Name your new database **my_database**



COURSE PLAN

	AM		PM
10:00 10:30	Welcome & Install-fest!	14:00 14:30	Commenting Code
10:30 11:00	Fundamentals of Databases and SQL	14:30 15:00	Functions in SQL
11:00 11:30	Your first query!	15:00 15:45	Unions & Joins
11:30 11:45	Break	15:45 16:00	Break
11:45 12:15	Filtering in SQL	16:00 16:30	Unions & Joins
12:45 13:15	Aggregation in SQL	16:30 17:00	Bringing it all together!

GENERAL ASSEMBLY

APPLYING FUNCTIONS IN SQL

COMMENTING CODE

COMMENTING CODE

- Headers are a great way to keep a history of why the SQL was built and who requested changes that have been made.
- Line-by-line usually works for calculations, and multi-line is a way to document more complicated processes and the reasoning behind them.

```
-- Basic commenting
    This works but is not ideal

/* Multiple Line comment is more universal */
```

COMMENTING CODE

- Headers are a great practice to adopt.

```
/**********************
** NAME: Name of report
** DESC: Description of report
** AUTH: Name of Author
** REQ: Name of requester
** DATE: Date report published
************************
**Change History
************************
** Version| Date     | Author |Description
**---------------------------------------
** 1.1        |10/15/16|Pat Doe| Description of change
*****************************/
```

GUIDED PRACTICE: MATH FUNCTIONS

MATH FUNCTIONS

- Math functions return calculated values from your data.

- For many of these functions, you will need to include a GROUP BY clause, or else the function won't know which rows to select.
- The default will be to use all rows, but some functions are not able to do this.

MATH FUNCTIONS

- COUNT:
  - Returns the number of rows that matches some specified criteria.
  - If criteria includes only a column name, returns the number of non-NULL values in that column.

  - Syntax:

    ```
    COUNT(field)
    ```

  - Example:

    ```
    SELECT store, COUNT(revenue)
    FROM sales
    GROUP BY store
    LIMIT 100
    ```

MATH FUNCTIONS

- AVERAGE:
  - The AVG() function returns the average value of a numeric column.
  - Syntax:

    AVG(field)

  - Example:

    SELECT store, AVG(revenue)

    FROM sales

    GROUP BY store

    LIMIT 100

MATH FUNCTIONS

- MIN: The MIN() function returns the smallest value of the selected column.

  - Syntax:

    ```
    MIN(field)
    ```

  - Example:

    ```
    SELECT store, MIN(revenue)
    FROM sales
    GROUP BY store
    LIMIT 100
    ```

MATH FUNCTIONS

- MAX: The MAX() function returns the largest value of the selected column.

  - Syntax:

    MAX(field)

  - Example:

    SELECT store, MAX(revenue)
    FROM sales
    GROUP BY store
    LIMIT 100

MATH FUNCTIONS

- SUM: This function is used to find the sum of a field in various records.

  - Syntax:

    SUM(field)

  - Example:

    SELECT store, SUM(revenue)
    FROM sales
    GROUP BY store
    LIMIT 100

APPLYING FUNCTIONS IN SQL

- Now that you know how to create new calculations in SQL you can also rename them!
- Simply add AS [new variable name] after the original variable or calculation.

```
SELECT store,
COUNT(revenue) AS count_sales,
AVG(revenue) AS avg_sales,
MIN(revenue) AS min_sales,
MAX(revenue) AS max_sales,
SUM(revenue) AS sum_sales
FROM sales
GROUP BY store
LIMIT 100
```

INDEPENDENT PRACTICE

APPLYING FUNCTIONS IN SQL

EXERCISE

Q1) What is the total of number of bottles sold (bottle_qty) and the total cost of sales (revenue) in our dataset?

Hint: Use SUM on sales table

Q2) How many stores are there by store_status, i.e. how many are active and how many are inactive?

Hint: Use GROUP BY on stores table

Q3) What is the average case_cost and average shelf_price by item_description for only category_name 'SCOTCH WHISKIES'?

Hint: Use AVG(), WHERE and GROUP BY on products table

GUIDED PRACTICE: STRING FUNCTIONS

STRING FUNCTIONS

- We will begin with string functions.
- Keep in mind that different SQL dialects may vary on syntax, but the sentiment is the same; it is easy enough to find a cross-reference for the function and vendor you may be using.

- Review a basic query as our starting point:
  - SELECT item, description
    FROM sales
    LIMIT 100

STRING FUNCTIONS

- Concatenation: Combines strings together.

  - Syntax:

    field1 || field2 || string || …

  - Example:

```
SELECT item || ' – ' || description
FROM sales
LIMIT 100
```

STRING FUNCTIONS

- Changing case:
  - LOWER: Converts a field or expression to lowercase.
    - Syntax: LOWER(field)
  - UPPER: Converts a field or expression to uppercase.
    - Syntax: UPPER(field)

  - Example:

```
SELECT UPPER(description), LOWER(category_name)
FROM sales
LIMIT 100
```

GUIDED PRACTICE: DATE FUNCTIONS

DATE FUNCTIONS

- It can be difficult to get into learning about dates.
- Syntax can be vastly different depending on the product.
  - For example, IBM’s current query tool uses a TIMESTAMPDIFF function, whereas PGADMIN uses a simple AGE function to the same effect.
- The best approach is to have an overall understanding of what DATES can do, and have your particular vendor's DATE syntax documentation (usually found online) close by.

DATE FUNCTIONS

- CURRRENT_DATE: brings back the current date from the system.
  - SYNTAX
    - CURRENT_DATE
  - EXAMPLE

```
SELECT item, revenue, date, CURRENT_DATE
FROM sales
LIMIT 100
```

DATE FUNCTIONS

- Days between dates: returns the difference between two dates.
  - SYNTAX
    - JULIANDAY(date1) – JULIANDAY(date2)
  - EXAMPLE

```
SELECT item, revenue, date, CURRENT_DATE,
JULIANDAY(CURRENT_DATE) - JULIANDAY(date)
FROM sales
LIMIT 100
```

GENERAL ASSEMBLY

QUERYING LARGE DATABASES IN SQL

LEARNING OBJECTIVES

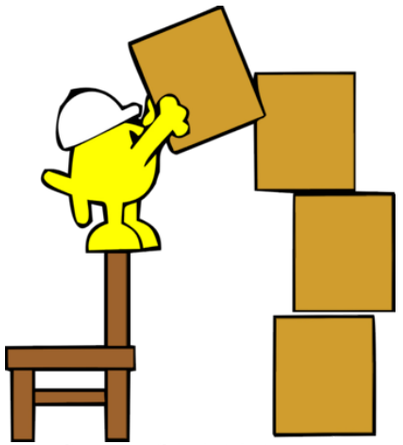
- Apply UNION to tables and columns for merging and further aggregation of data.
- Use JOINS to create relationships between tables to obtain data.

INTRODUCTION: UNIONS AND JOINS

QUERYING LARGE DATABASES IN SQL

- Apply **UNION** to tables and columns for merging and further aggregation of data.

Stacking!



Credit: ChristArt.com

- Use **JOINS** to create relationships between tables to obtain corresponding data.

Zipping!

Credit: http://femfusionfitness.com/zipping-up/

SAMPLE CODE FOR UNIONS

- What is the difference between a UNION and a JOIN?
  - UNION merges data, whereas a JOIN uses fields from different tables in the final results.

DEMO: SAMPLE CODE FOR UNION

SAMPLE CODE FOR UNIONS

- Let's look at some simple mock syntax for UNION:

```
SELECT field1
FROM table1
UNION
SELECT field1
FROM table2
```

GUIDED PRACTICE: UNIONS

UNIONS

- Now we can build three different examples of UNIONS: A table UNION, a column UNION, and a combination of the two.

- SAMPLE TABLES:

Employees1

id	first_name	last_name	current_salary
2	Gabe	Moore	50000
3	Doreen	Mandeville	60000
5	Simone	MacDonald	55000

Employees2

id	first_name	last_name	current_salary
7	Madisen	Flateman	75000
11	Ian	Paasche	120000
13	Mimi	St. Felix	70000

UNIONS

- A Table UNION includes selections from different tables:

```
SELECT *
FROM Employees1
UNION
SELECT *
FROM Employees2
```

id	first_name	last_name	current_salary
2	Gabe	Moore	50000
3	Doreen	Mandeville	60000
5	Simone	MacDonald	55000
7	Madisen	Flateman	75000
11	Ian	Paasche	120000
13	Mimi	St. Felix	70000

UNIONS

- Table Union:

```
SELECT *
FROM FY17P
UNION
SELECT *
FROM FY18P
```

fy numeric	pd numeric	store_na... character	week1 numeric	week2 numeric	week3 numeric	week4 numeric
18	2	SAN DIEGO	754.959568	803.757709	5687.76688	219.527173
18	2	DALLAS	103.293083	217.776958	648.601179	4531.9008
18	2	SEATTLE	869.439661	595.887082	954.010546	746.064601
17	2	SAN DIEGO	000.715479	741.061154	400.657862	112.872637
17	2	DALLAS	266.472034	698.658564	045.834247	762.222799
17	2	PORTLAND	569.060961	415.029643	67.8186051	72.2430083
17	2	PORTLAND	351.793961	485.721046	425.021263	306.720805
18	2	PORTLAND	729.988537	06.4388102	150.983333	198.229664
18	2	PORTLAND	301.496764	3956.17356	127.618963	779.973338
18	2	SEATTLE	789.640585	194.083103	282.101838	744.869032
18	2	PORTLAND	002.225137	125.229579	245.442398	7059.23758
17	2	SEATTLE	692.497933	181.314618	641.401526	082.684282
17	2	PHOENIX	72.3659485	900.969922	737.209742	471.024603
18	2	VANCOUV...	913.520693	583.003705	768.463565	214.983293
17	2	VANCOUV...	293.769182	447.159002	092.626357	025.499113

UNION RULES

- UNION removes duplicates ↔ UNION ALL allows duplicates.

```
SELECT *
FROM FY17p
WHERE store_name = 'PORTLAND'
UNION
SELECT *
FROM FY17p
WHERE store_name = 'PORTLAND'
```

```
SELECT *
FROM FY17p
WHERE store_name = 'PORTLAND'
UNION ALL
SELECT *
FROM FY17p
WHERE store_name = 'PORTLAND'
```

	fy numeric	pd numeric	store_name character (80)	week1 numeric	week2 numeric	week3 numeric	week4 numeric
1	17	2	PORTLAND	)43.511757	.18.030799	'4.4964829	)06.857039
2	17	2	PORTLAND	351.793961	485.721046	425.021263	306.720805
3	17	2	PORTLAND	569.060961	415.029643	57.8186051	'2.2430083

	fy numeric	pd numeric	store_name character (80)	week1 numeric	week2 numeric	week3 numeric	week4 numeric
1	17	2	PORTLAND	351.793961	485.721046	425.021263	306.720805
2	17	2	PORTLAND	569.060961	415.029643	57.8186051	'2.2430083
3	17	2	PORTLAND	)43.511757	.18.030799	'4.4964829	)06.857039
4	17	2	PORTLAND	351.793961	485.721046	425.021263	306.720805
5	17	2	PORTLAND	569.060961	415.029643	57.8186051	'2.2430083
6	17	2	PORTLAND	)43.511757	.18.030799	'4.4964829	)06.857039

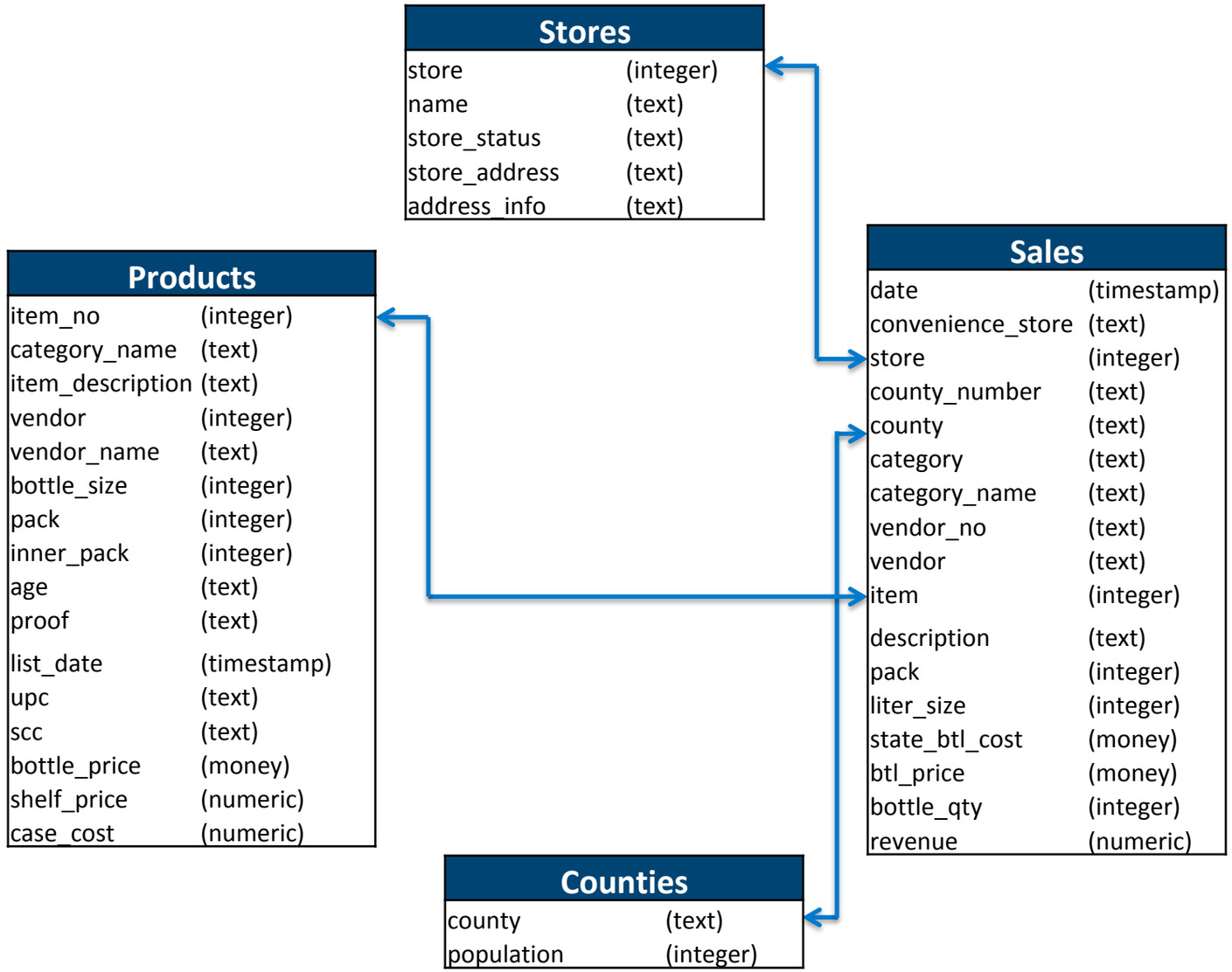
INTRODUCTION: JOINS

QUERYING LARGE DATABASES IN SQL

Database schema model

If possible, ask the database administrator to give you the ERD (Entity-Relationship Diagram) model.

Example for Iowa database:



JOIN SYNTAX

- When you create a JOIN, each TABLE can have an ALIAS and each FIELD is connected to the TABLE via the ALIAS.

table1 a	→	a.field1	table1 uses the alias a
table2 b	→	b.field1	table2 uses the alias b

- After this statement is the table to which you want to link.

- Specify the dimension on which you want to link tables, which is done with ON a.column_name = b.column_name

DEMO: JOIN SYNTAX

JOINS

- Here is an example of the JOIN syntax.
- Notice the store aliases and the field names. This is to indicate which tables the fields are coming from.

```
SELECT
    a.store,
    a.description,
    b.store_status
FROM
    sales a
JOIN
    stores b
ON a.store = b.store
LIMIT 100
```

Aliases are a good practice to keep track of where fields are coming from.
If there is more than one column named the same from different tables, the query will fail without specifying which table it comes from.

The connecting variables don't have to be named the same.
The join will still work as long as the data contained in them is compatible.

INDEPENDENT PRACTICE: JOINING SALES TO PRODUCTS

JOINING SALES TO PRODUCTS

EXERCISE

- There's new regulation being enforced that requires all liquor sales to display its proof next to each transaction made.
- You know that column proof is in table products but you need to show it along all the sales details.
  - How can you build a query with a join that will show the proof for each item sold in the sales table?
  - LIMIT 100 to make the query run faster.

DEMO: JOINING MULTIPLE TABLES

JOINING MULTIPLE TABLES

- Here is an example of joining multiple tables together. Notice the code repeats itself.
- Tip: It can be helpful to sketch and wireframe how your JOINS will relate on a piece of paper before coding.

```
SELECT
b.field1, a.field2, a.field3, c.field4
FROM table1 a
JOIN table2 b
ON a.field1 = b.field1
JOIN table3 c
ON a.field1 = c.field1
LIMIT 100
```

Base query
Can select fields from any table in the query

Add a table and link

Add another table and link, etc

INDEPENDENT PRACTICE: JOINING MULTIPLE TABLES

JOINING MULTIPLE TABLES

EXERCISE

- Using Sales as the primary table, create links to all of the other tables in the Iowa liquor database.
  - Result should be one query with several JOINS and bring back:

- item from the sales table
- name from the stores table
- case_cost from the products table
- population from the counties table

- Limit to 100
- Before proceeding, practice wireframing your JOINS out on a piece of paper.

Hint: Use the ERD schema from the beginning of the class

LET'S BRING IT ALL TOGETHER!

LET'S BRING IT ALL TOGETHER!

- You are now working with the CEO of the Iowa Liquor Stores company and he would like to ask you the following questions:

Q1) Could you show me a list of sales with the average case cost by item but only the ones with proof above 80?

EXERCISE

Q2) I think that our sales table is great, however could I also have a new column where I can see the store number and name together e.g. '3692 - Wilkie Liquors'?

Q3) We have been stocking some products for a long time now. Are you able to tell me, by category name, what is the maximum number of days we've had a product listed for? (e.g. days since list date)

- Limit to 100
- Before proceeding, practice wireframing your JOINS out on a piece of paper.

Hint: Use the ERD schema from the beginning of the class

CONCLUSION

CONCLUSION

- Understand the differences between different data management systems for SQL databases
- Create basic SQL queries by filtering, sorting, aggregating and commenting
- Use advanced SQL commands [GROUP BY, HAVING] and filtering data [=,!=,>,<, IN, NOT IN, LIKE and BETWEEN]
- Apply SQL aggregate functions [MIN, MAX, SUM, AVG, COUNT]
- Apply string, date, and math functions to manipulate how data is presented
- Using UNION and JOINS for merging data