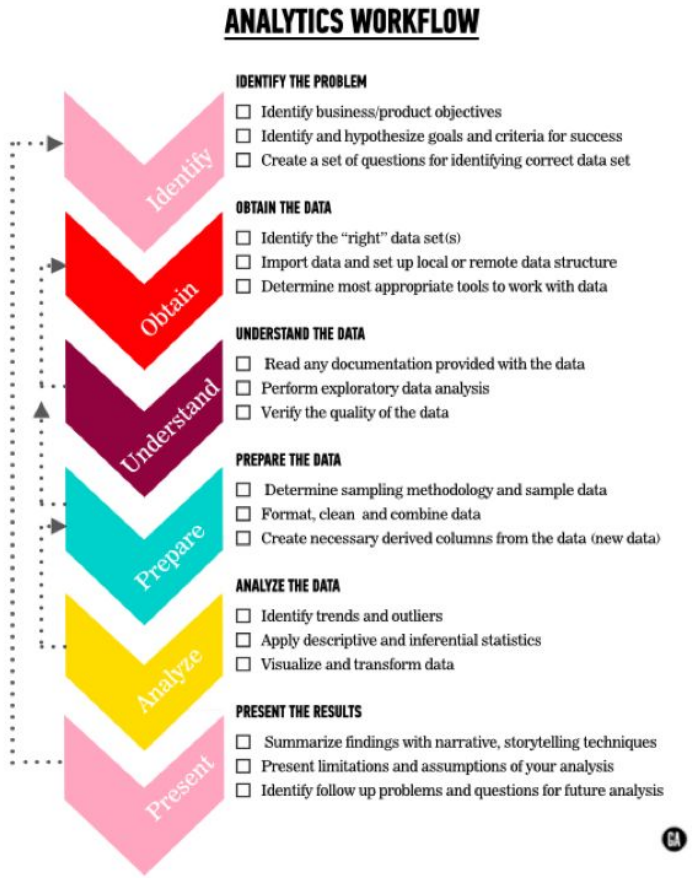
INTRODUCTION TO SQL

Anahita Bahri

Analytics Workflow

1. Identify the problem
2. Obtain the data
3. Understand the data
4. Prepare the data
5. Analyze the data
6. Present the results



Obtaining your data

Often, obtaining data means accessing a SQL server using the SQL language to get the data you need

OBTAIN THE DATA

- ☐ Identify the "right" data set(s)
- ☐ Import data and set up local or remote data structure
- ☐ Determine most appropriate tools to work with data

Why SQL?

Do you store raw data in Excel spreadsheets, or is there another way?

Why Databases?

- A database is a collection of data and metadata
- How data is often stored in a database at point of capture
    - We wouldn’t expect software our company uses to collect data to store that data directly into Excel or a CSV
    - Databases are used as a repository of information

Two types of databases

- **Relational databases** are well-defined table structures of data with defined relationships between those tables.
  - Relational databases require a lot of planning because they are relatively rigid (tables and relationships are well-defined and all data conforms)
  - We will be focused on relational databases in this unit. They are much more common and much older (since the 1970s!)

- **Document-based databases** are collections of object that don't necessarily have a standard definition.
  - Document-based systems (often called "NoSQL") are quickly gaining popularity, but are still (relatively) immature
  - Unstructured data: word docs, PDFs, audio files, videos, presentations, etc

What is SQL? What can you do with it?

- Relational databases typically use a language called SQL, the **Structured Query Language.**
- SQL is good at fetching data and is most typically used to retrieve information from databases and combine it together to create reports.
- In this workshop, we will be learning and using SQL to access data in a relational database.

Why is SQL in-demand?

- Think of just about any organization from big banks, to hospitals, to app startups…
- All companies rely on data and need to organize and understand the information in a relevant way.
- They are always going to need a database professional.

Who uses SQL?

- Everyone who uses a relational database uses SQL
- To name a few:
  - Back-end developers
  - Data scientists
  - Data analysts
  - Data-driven marketers
  - Data-driven account managers

Types of SQL implementations

- There are a number of implementations of relational databases, including:
  - PostgreSQL (what we will be using)
  - MySQL
  - MS SQL
  - Oracle
- Each has their own pros and cons on top of supporting standard (ANSI) SQL as a query language

Why PostgreSQL?

- We will be using PostgreSQL for this class, because
  - It's popular
  - It's free and open-source
  - It's powerful
  - The skills we will learn are largely applicable to other implementations
  - We have access to data on the GA server :)

I Do

SQL Intro: Shazam Example

- I have a giant database that includes thousands of columns and millions of rows. I don't need all these columns and rows! (see below for 7/100+ col names)
- I want the following from the table called 'ShazamData':
  - Columns: Country, Genre, Artist
  - Special conditions:
    - Genre is either hip-hop or electro-pop
    - Country is United States
- Can you think of any problems I may come across?

DATE	CITY	COUNTRY	GENRE	ARTIST	SONG
11/8/2016	BOSTON	UNITED STATES	ELECTROPOP	MGMT	LITTLE DARK AGE

Connect to the AWS Server using PgAdmin

Please connect to the region specific server:

- host: analyticsga-east2.c20gkj5cvu3l.us-east-1.rds.amazonaws.com
- port: 5432
- username: analytics_student
- password: analyticsga

Connecting to AWS Server

- Let's try this together
  - Open PgAdmin
  - Connect to the GA Analytics Server
  - Locate the Iowa Liquor Sales Database
    - Schemas → Public → Click Tables (4)
    - Click the SQL magnifying glass at the very top of your screen
  - Explore the tables in this database

SQL Queries

```
SELECT *
FROM products
LIMIT 100;
```

- What does * mean?
- What does the “FROM products” mean?
- What does the LIMIT do?
- I see the same category name repeated many times. Why is this? I’d like to see all the unique values of category name.

Select Distinct

- There is a special query type that summarizes or describes our data.
- By using SELECT DISTINCT instead of just SELECT, we get back the unique combinations of the variables or columns we specify

- Let's try to get the unique category names.
  - SELECT DISTINCT category_name
    FROM products;

- What's weird here?

Null values

- What is a NULL value? What does it represent?
  - A null value represents the absence of data
  - It is different from 0. 0 is a data point that says the data point is 0.
  - Null says we don't know what the data point is

- By using SELECT DISTINCT and not removing NULLs, the NULL value itself will count as a unique value but it does not actually represent a product name

Select Distinct with Null Values

Updated query for unique category names, excluding the null values.

```
SELECT DISTINCT category_name
FROM products
WHERE category_name is not null;
```

Filtering Operator

- You can use the following operators with the where statement
  - =
  - != (Not equal to)
  - >, >=
  - <, <=
  - LIKE (Case sensitive)
  - ILIKE (Case insensitive)
  - BETWEEN
  - IS NULL
  - IS NOT NULL
  - IN (INCLUDE)
- Numeric / Integer data type? You don’t need single quotes for the data.
- Text or other data type? You need single quotes for the data.

Filtering Operator

- Numeric / Integer data type (excluding money data type)? You don't need single quotes for the data.
  - WHERE age > 50 ✓
  - WHERE age > '50' OK but not needed
- Text or other data type? You need single quotes for the data.
  - WHERE name = 'Anahita' ✓
  - WHERE name = Anahita ✖

Order of Operation

- Order matters in these queries. Typically, the order is:
  - SELECT
  - FROM
    - JOIN (if you are joining)
  - WHERE
  - GROUP BY
  - HAVING
  - ORDER BY
  - LIMIT

We Do

SQL Queries

- Let's answer the following questions using SQL queries by writing and executing SQL queries.
- When confused about which table to pull data from, take a look at the graphical query builder.
    - Which items come in packs larger than 12?
    - Which items have a case cost of less than $70?
    - Which items come in packs larger than 12 AND have a case_cost of less than $70?
    - Which categories have a proof of 85 or more?
    - Which items are in the whiskey (or whiskies) category?

SQL Queries

- Which items come in packs larger than 12?
  - SELECT DISTINCT item_description, pack
    FROM products
    WHERE pack > 12
    ORDER BY pack DESC;

- Which items have a case cost of less than $70?
  - SELECT item_description, case_cost
    FROM products
    WHERE case_cost < 70 and case_cost != 0
    ORDER BY case_cost;

SQL Queries

- Which items come in packs larger than 12 AND have a case_cost of less than $70?
  - SELECT item_description, pack, case_cost
    FROM products
    WHERE pack > 12 and case_cost <70;

- Which categories have a proof of 85 or more?
  - SELECT DISTINCT category_name, cast(proof as integer)
    FROM products
    WHERE cast(proof as integer) >= 85 and category_name is not null;

SQL Queries

- Which items are in the whiskey (or whiskies) category?
  - SELECT DISTINCT item_description, category_name
    FROM products
    WHERE category_name like '%WHISK%';

You Do

Filtering Queries

- Answer the following questions:
  - Which items are over 90 proof?
  - Which items have a case cost of less than $60?
  - Which items are in either Single Malt Scotches or Canadian Whiskies categories? Hint: you don't have to use LIKE here, but explore how Single Malt Scotches + Canadian Whiskies are formatted / spelled.
  - Which items are in the whiskey or whiskies category OR have a proof over 85?

Answers

- Which items are over 90 proof?
  - SELECT DISTINCT item_description
    FROM products
    WHERE cast(proof as integer) > 90;

- Which items have a case cost of less than $60?
  - SELECT DISTINCT item_description, case_cost
    FROM products
    WHERE case_cost < 60;

Answers

- Which items are either Single Malt Scotches or Canadian Whiskies (based on category name)?
  - SELECT DISTINCT item_description, category_name
    FROM products
    WHERE category_name = 'SINGLE MALT SCOTCH'
        OR category_name = 'CANADIAN WHISKIES';

- Which items are in the whiskey or whiskies category OR are over 85 proof?
  - SELECT DISTINCT item_description, category_name, cast(proof as numeric)
    FROM products
    WHERE cast(proof as numeric) > 85 AND category_name like '%WHISK%';

Intermediate SQL Queries // Intro to GROUP BY and HAVING

Aggregation in SQL

- A GROUP BY clause is needed when you use aggregate functions in the SELECT statement. It is used to group the result-set by one or more columns.
- HAVING clause help us work with aggregate data.

Count

- What do you think this query will do?

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

Count()

- The COUNT() function is an example of an aggregate function and can be used to aggregate or summarize our data.
- When you have an aggregate function in your SELECT statement, **you need a GROUP BY statement including everything in your SELECT statement except for aggregate function**.
- While that might be useful, running aggregate functions for groups of values tends to be more interesting.
- For example: how would I find out how many items are available *per vendor name*?

Count()

- How would I find out how many items are available *per vendor name*?

```
SELECT vendor_name, COUNT(item_description)
FROM products
GROUP BY vendor_name;
```

Aggregate Functions

- Count()
- Min()
- Max()
- Sum()
- Avg()

HAVING clause

- What if we wanted to know which categories have an average proof higher than 75?

HAVING clause

- What if we wanted to know which categories have an average proof higher than 75?
  - SELECT category_name, avg(cast(proof as numeric))
    FROM products
    WHERE AVG(CAST(proof as numeric)) > 75
    GROUP BY category_name;

- Run it. What's wrong?

HAVING clause

- A WHERE clause will NOT work because it only operates on data from the data and we need to filter data performed by the aggregate function.
- So, aggregate functions are not allowed in WHERE!
- Instead we can use a HAVING clause, which is very similar to a WHERE clause but works on filtering data produced *within the query*.

- Try again: which categories have an average proof higher than 75?

HAVING clause

```
SELECT category_name, AVG(CAST(proof as numeric))
FROM products
GROUP BY category_name
HAVING AVG(CAST(proof as numeric)) > 75;
```

You Do

Independent Practice

- Answer the following questions and remove nulls when necessary:
  - How many items does each category have?
  - Which categories have more than 100 items? I'd like to have a column including the count too.
  - What is the average bottle size per category of whiskey?
  - What is the average state_btl_cost per vendor? Add a condition for avg state_btl_cost to be over 10.

Independent Practice

- How many items does each category have?
  - SELECT category_name, COUNT(item_no)
    FROM products
    WHERE category_name IS NOT NULL
    GROUP BY category_name;

- Which categories have more than 100 items?
  - SELECT category_name, COUNT(item_no)
    FROM products
    WHERE category_name IS NOT NULL
    GROUP BY category_name
    HAVING COUNT(item_no) > 100;

Independent Practice

- What is the average bottle size per category of whiskey?
  - SELECT category_name, AVG(bottle_size)
    FROM products
    WHERE category_name LIKE '%WHISK%'
    GROUP BY category_name;

- What is the average state_btl_cost per vendor? Add a condition for avg state_btl_cost to be over 10.
  - SELECT vendor, AVG(CAST(state_btl_cost as numeric))
    FROM sales
    WHERE vendor IS NOT NULL
    GROUP BY vendor
    HAVING AVG(CAST(state_btl_cost as numeric)) > 10;

Mathematical Operations

Mathematical Operations

- Can be carried out within a row in SQL using operators (+, -, *, /) or using other mathematical functions (floor, round, sqrt, etc.)
- **Example**: Let's try to find the price per bottle for each item.

Mathematical Operations

- The price per bottle for each item:

```
SELECT description, total / bottle_qty as PriceBtl
FROM sales
LIMIT 100;
```

Queries with mathematical functions

- Add a column named “difference” to the products table that calculates the difference between shelf price and bottle price for each product.
- Hint: Make sure shelf price and bottle price are the *same* data type.

Queries with mathematical functions

- Add a column named “difference” to the products table that calculates the difference between shelf price and bottle price for each product.
- Hint: Make sure shelf price and bottle price are the *same* data type.

```
SELECT *, (shelf_price - (cast(bottle_price as numeric))) as difference
FROM products;
```

You Do

Independent Practice

- Answer the following questions and remove nulls when necessary:
  - Calculate the price (using bottle price) per mL (using bottle size) for each product
  - Which 5 products cost the most per mL?

Independent Practice

- Calculate the price (using bottle price) per mL (using bottle size) for each product
  - SELECT bottle_price / bottle_size AS price_per_ml
    FROM products
    WHERE bottle_size IS NOT NULL AND bottle_size != 0;

- Which categories have more than 100 items?
  - SELECT item_description, category_name, bottle_price / bottle_size as price_per_ml
    FROM products
    WHERE bottle_price IS NOT NULL AND bottle_size IS NOT NULL AND bottle_size != 0
    ORDER BY price_per_ml DESC
    LIMIT 5;

Advanced SQL: Joins

Joins

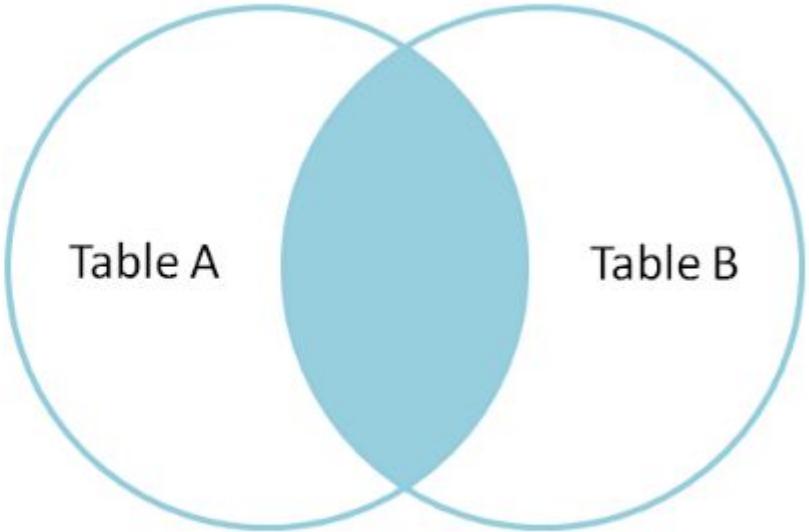
- Assume we have the following two tables. **Table A** is on the left, and **Table B** is on the right. We'll populate them with four records each.

id	name	id	name
1	Pirate	1	Rutabaga
2	Monkey	2	Pirate
3	Ninja	3	Darth Vader
4	Spaghetti	4	Ninja

[redacted]

```
SELECT * FROM TableA
INNER JOIN TableB
ON TableA.name = TableB.name

id  name       id   name
--  ----       --   ----
1   Pirate     2    Pirate
3   Ninja      4    Ninja
```



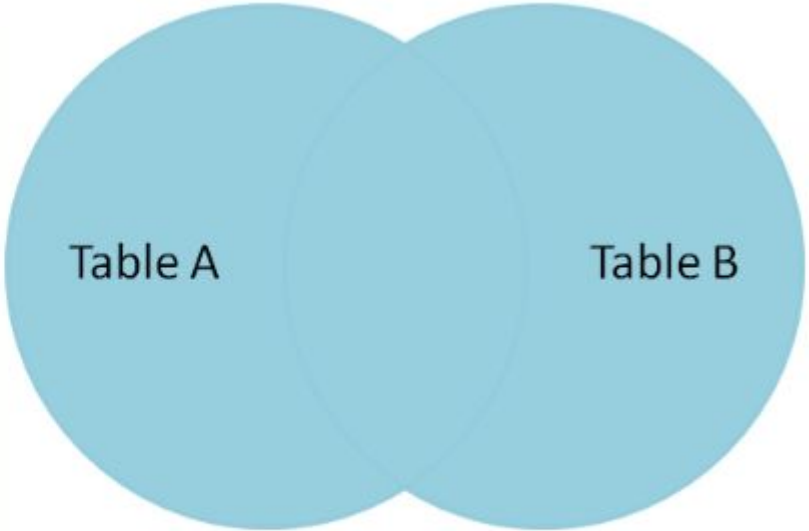
Inner join produces only the set of records that match in both Table A and Table B.

```
id name       id   name
-- ----       --   ----
1  Pirate     1    Rutabaga
2  Monkey     2    Pirate
3  Ninja      3    Darth Vader
4  Spaghetti  4    Ninja
```

Full Outer Join

```
SELECT * FROM TableA
FULL OUTER JOIN TableB
ON TableA.name = TableB.name
```

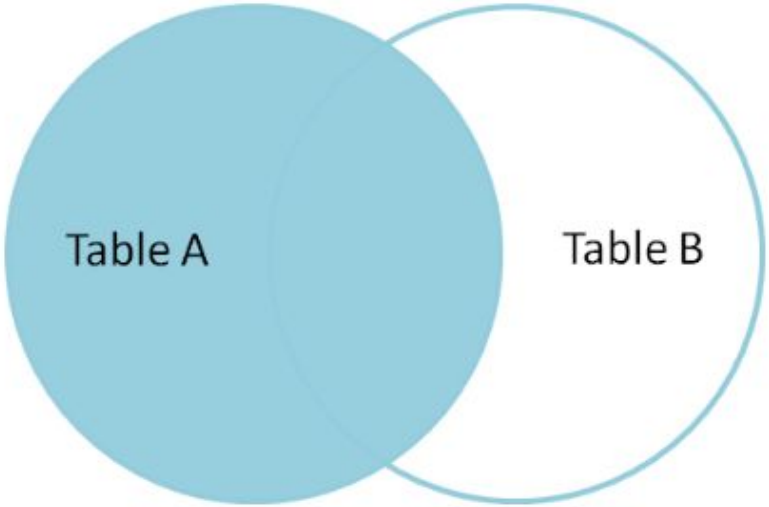
id	name	id	name
1	Pirate	2	Pirate
2	Monkey	null	null
3	Ninja	4	Ninja
4	Spaghetti	null	null
null	null	1	Rutabaga
null	null	3	Darth Vader



Full outer join produces the set of all records in Table A and Table B, with matching records from both sides where available. If there is no match, the missing side will contain null.

```
SELECT * FROM TableA
LEFT JOIN TableB
ON TableA.name = TableB.name
```

id	name	id	name
1	Pirate	2	Pirate
2	Monkey	null	null
3	Ninja	4	Ninja
4	Spaghetti	null	null



Left join produces a complete set of records from Table A, with the matching records (where available) in Table B. If there is no match, the right side will contain null.

id	name	id	name
1	Pirate	1	Rutabaga
2	Monkey	2	Pirate
3	Ninja	3	Darth Vader
4	Spaghetti	4	Ninja

We Do

Joins Practice

- Let's find the top 5 store sales with its store name.
  - Store *name* is found in the stores table.
  - Store sales = sum of total in sales table.
    - Since we will include a sum (aggregate function) in our SELECT statement, what will we need to add to our query?
  - We need to join the sales table to the stores table.
    - What can be found in both tables?

Sample Solution Query

```
SELECT stores.name, sum(sales.total)
FROM sales
LEFT JOIN stores
ON sales.store=stores.store
GROUP BY stores.name
ORDER BY sum(sales.total) DESC
LIMIT 5;
```

```
SELECT b.name, sum(a.total)
FROM sales as a
LEFT JOIN stores as b
ON a.store=b.store
GROUP BY b.name
ORDER BY sum(a.total) DESC
LIMIT 5;
```

You Do

Practice

- Which unique categories, other than whiskies, were sold in Mason City, IA?

Answers

- Which unique categories, other than whiskies, were sold in Mason City, IA?
  - SELECT DISTINCT a.category_name, b.store_address
    FROM sales a
    LEFT JOIN stores b USING (store)
    WHERE a.category_name not like '%WHISKIES%' and b.store_address like '%Mason City%'

Advanced SQL: CASE Statement

CASE Statements

- CASE statements are a lot like “if” statements in a programming language: they allow you return different values given the values within the database.
- CASE statements are a great way to group ranges of data into categorical groups, as well will see.
- For example, say you have a database of student test scores. You could use a case statement to break up a numerical “grade” column into 5 letter grades: A, B, C, D, F

CASE Statements

- CASE statements take the value of one variable and map it to another value. This behavior can be similar to “if” statements or switches in other languages.

- **Example**: Say you have a list of world cities and their populations. You could classify each city as either small, medium, or large based on population ranges you define to fit into each category.

- CASE statements have a special syntax in SQL

CASE Statement syntax

```
SELECT something,
        CASE
                WHEN condition THEN value
                WHEN condition THEN value
                [...]
                [ELSE value]
        END AS casecolname
FROM table;
```

We Do

CASE Statements in Queries

- **Example**: We want to classify the counties of Iowa as either small, medium, or large. For the purposes of this example, say large is over 200,000, medium is between 40,000 and 200,000 and small is anything lower than 40,000.

Sample Solution Query

- **Example**: We want to classify the counties of Iowa as either small, medium, or large. For the purposes of this example, say large is over 200,000, medium is between 40,000 and 200,000 and small is anything lower than 40,000.
  - SELECT county, population,
    CASE
    WHEN population >= 200000 THEN 'large'
    WHEN population >= 40000 THEN 'medium'
    ELSE 'small'
    END AS county_size
    FROM counties;
  - You can add an order by, if you'd like.
  - How about counting how many counties are in each county_size?

Advanced SQL: Subqueries

Subqueries

- A subquery is a query within a query. This allows you to use the results of a query in another query.
- The general structure looks like this:
  - SELECT something
    FROM (
        SELECT something_else
        FROM somewhere
) AS temp
    GROUP BY something (if applicable);

Subqueries

```
SELECT something
FROM (
        SELECT something_else
        FROM somewhere
        ) AS temp
GROUP BY something (if applicable);
```

- **Note:** Here we see that where we normally have a table, we can supply an entire query between parentheses. The outer query then treats the results of the inner query as if it were another table. While these are very simple queries, each SELECT query could be as complex as you like.
- **Note**: you MUST give an alias name to the inner query with the AS keyword, even if it you never refer to the inner query by this alias.

Subquery Challenge

- Let's aggregate the following new classifications to get a count of how many counties fit in each group.
- The following will serve as our inner query:
  - SELECT county, population,
    CASE
    WHEN population >= 200000 THEN 'large'
    WHEN population >= 40000 THEN 'medium'
    ELSE 'small'
    END AS county_size
    FROM counties;
- Let's draw out what our ideal table would be . . .

Sample Solution Query

```
SELECT county_size, count(*) AS n_county
FROM (
        SELECT county, population,
            CASE
                WHEN population >= 200000 THEN 'large'
                WHEN population >= 40000  THEN 'medium'
                ELSE 'small'
            END AS county_size
        FROM counties
        ) AS temp
GROUP BY county_size;
```

- **Note:** GROUP BY clause must be part of the outer query, not the inner query, since the aggregation is happening to the result of the inner query.

You Do

Practice Queries

- Use subqueries and CASE statements when possible.
  - What were the top 5 categories of liquor sold (based on number of sales) in the five most populous counties?
  - In our products table, there are many different kinds of whiskeys [category_name]. Create a column classifying whiskeys as 1 and other category_names as 0.
    - What's the percentage of whiskey categories in this table?

Answers

- What were the top 5 categories of liquor sold (based on number of sales) in the five most populous counties?
  - SELECT category_name, count(*) AS n
    FROM sales
    WHERE county IN (
    SELECT county FROM counties ORDER BY population DESC limit 5)
    GROUP BY category_name
    ORDER BY n DESC
    LIMIT 5;

Answers

- What's the percentage of whiskey categories in this table?
  - SELECT AVG(is_it_whiskey)
    FROM (
    SELECT *,
    CASE
        WHEN category_name LIKE '%WHISK%' THEN 1
        ELSE 0
    END AS is_it_whiskey
    FROM products) as temp

Answers

- What's the percentage of whiskey categories in this table?
  - SELECT
    AVG(CASE
        WHEN category_name LIKE '%WHISK%' THEN 1
        ELSE 0
    END)
    FROM products;

Next Steps

SQL Practice Moving Forward

- Practice the basics on Codeacademy
- Explore the Iowa Liquor Database with your own questions
- Redo the questions listed in this presentation *without* looking at the answers
- Explore the other databases on the GA server
- Want to showcase your SQL skills through a project? Think of big-picture questions or problems you can solve through the GA data available to you.
    - Ex: An alcohol brand of your choice is looking to further expand in Iowa. In which counties are they selling the most / least? What are they selling? How can they improve? What trends do you see?

Slides here: goo.gl/oxDcv8

GOOD NIGHT!

- abahri@bu.edu
- @anahitabahri
- LinkedIn