GENERAL ASSEMBLY - BOOTCAMP
SQL WORKSHOP

INTRODUCTION

Purpose of SQL
- Relational database
- Purpose
  1. Maintain data Integrity
  2. Speed of operations under large data quantity
     2.1. Operators
     2.2. Functions/formulas
  3. Encourages collaboration

Name	DOB
John Travolta	9/18/56
Amy	July 7, 1984

- Above users do not follow the same convention

What is SQL?
- Selective data selection on limited data space
- Data search would move in one direction and restart from begging on new query
- Relational calculus functions to structure data in a way that allowed data to be searched in a non-linear way
- Human interface for generating calculus functions
  - PostGris
  - SQLLite
  - Additional features over SQL
  - All except the same way in which SQL is implemented
- RDBMS
  - Relational database management system

MySQL and MySQL Workbench

MySQL
- No GUI - Accept SQL commands and sends data to Work bench
- Specify address and port number to store and run database
- IP Address:
  - Default: 127.0.0.1
  - “local host”’
- Port:
  - Each program runs on a separate port
  - 3306: Default port
- Various work benches can request data

Work Bench
- Package SQL commands for MySQL Server
- Specifies SQL commands and requests data from MySQL
- IP: 127.0.0.1
- Port: 58
- GUI to allow communication to MySQL
- “Test connection”
  - Test the connection between IP and

SCHEMA

- Separation/segmentation of data for various data sets
  - Projects
  - Themes
  - etc
- Default collation
  - utf8 - default

COMMANDS & PROCESS

Data Types

varChar - Variable character of 255 characters
int - Integer
NULL - Nothing
NOT NULL - Something

1. Creating a table

```
CREATE TABLE 'users'(
    'column' datatype default data data entry
    PRIMARYKEY = 'column'
)
```

2. Inserting data into table

```
INSERT INTO 'nameOfTable: VALUES ('column1', 'column2',...,'columnN')
```

3. Query information on tables

a. Commands

i. Order in which command are specified matters

1. Location of command does not matter

i. SELECT columnName1, columnName2 FROM tableNAME;
ii. SELECT * FROM tableNAME;
    1. Select all columns irrespective of repetitions
iii. ORDER BY
    1. SELECT * FROM tableNAME ORDER BY columnName DESC;
        a. DESC - Descending order
        b. Default - Ascending order
        c. Orders based on data type
iv. WHERE IN/LIKE
    1. SELECT * FROM tableNAME WHERE columnName = datatype
        a. Filters data according to data entries
        b. Need to specify the data type e.g. WHERE gender = 'female'
        c. Quotes: varchar
        d. Non-quotes: column name
        e. Strict equality '='
            i. Find data strictly equal to specification

2. SELECT * FROM tableNAME WHERE columnName = datatype IN (dataType1, dataType2)
3. SELECT * FROM tableNAME WHERE columnName = datatype LIKE (dataType1%)
    a. % - wildcard e.g.
        i. 'New %'
        ii. '%y%

v. WHERE AND/OR
1. SELECT * FROM tableNAME WHERE columnName = datatype AND/OR columnName = dataType

vi. DISTINCT
1. SELECT DISTINCT columnName FROM tableNAME
    a. Removes duplicates and displays entries from specified column in table
    b. Returns distinct states and skips rows that were previously identified

b. Functions

1. Discoloring of text based on identification of special text
    a. Not to be thrown by this
2. SELECT columnN FROM tableNAME WHERE FUNCTION(arguments)
    a. Function arguments refer to the number of inputs required to execute the function
    b. Refer to SQL documentation
        i. https://docs.oracle.com/cd/B19306_01/server.102/b14200/functions001.htm
        ii. https://dev.mysql.com/doc/
        iii. Google - SQL CHEAT SHEET
    c. E.g
        i. LENGTH()
        ii. CONCAT_WS()
            1. SELECT CONCAT_WS(Separation character, First column to concatenate, Second column to concatenate)

3. LIMIT/OFFSET
    a. Define number of entries to display
        i. SELECT * tableNAME LIMIT n
        ii. Need to determine percentage and convert to integer value to call values under investigation

4. Descent a custom data set in descending order of its data type
    a. SELECT table1, count(*) AS variableName FROM table 2 GROUP BY table 3 ORDER BY variableName DESC;

5. SUM()
    a. SELECT SUM(columnName) FROM table;
        i. Summate all values from a column
    b. SELECT SUM(columnName) FROM table
       GROUP BY columnName2

6. WHERE and HAVING
    a. Having only works with aggregate functions
    b. SELECT songs.title, COUNT(*) AS total_instruments From

Exercise

1. List all Presidents
    1.1. SELECT * FROM presidents;
2. List only presidents who died before 1900
    2.1. SELECT first_name, last_name FROM presidents WHERE death < '1900-01-01'
3. List the unique first name of all presidents
    3.1. SELECT DISTINCT first_name FROM presidents LIMIT 0,1000
4. List the presidents in order of oldest to youngest - Specifically, the first 10
    4.1. SELECT * FROM presidents ORDER BY birth DESC LIMIT 10
5. Replace NULL data in AGE column with their actual age

RELATIONAL DATABASE MANAGEMENT

- Relational data
    - Data is related with each other in some way or the other. Data is interrelated.

Name	Email	Dogs name
John Travolta	john@travolta.com	Lenny
Amy Cruise	amy@cruise.com	Betty

- Problems with above,
    - Both customers can have the same name of dogs
    - Both customers are not unique as they can have the same name

- Solution
    - Adding columns may not solve the problem in the long term
    - Create a separate table that is liked to the primary customer

Dogs	Breed	Owner Name
Benji	Lenny	Carrie Fischer
Winnie	Betty	Carrie Fischer
Fido	Poodle	Carrie Fischer

    - Owner Name is *still* not unique

SOLUTION

- Add an identifier to each entsry that is intentionally unique

Customer ID (PK)	Name	Email
1	John Travolta	john@travolta.com
2	Amy Cruise	amy@cruise.com

- In the case where an entry is removed from the system, the ID remains the same
- Link additional tables to ID
  - Create a relationship between the ID and column
  - Specification of primary key is in a table that points to a specifically unique data point.
  - Define the primary key to be the column that uniquely identifies entries in the database and defines interrelation between data

Dogs	Breed	Owner Name	Customer ID (FK)
Benji	Lenny	John Travolta	1
Winnie	Betty	Amy Cruise	2
Fido	Poodle	John Travolta	1

- Foreign key
  - Defines and identifier that points out to a primary key on another table

DOG ID (PK)	Dogs	Breed	Owner Name	Customer ID
1	Benji	Lenny	John Travolta	1
2	Winnie	Betty	Amy Cruise	2
3	Fido	Poodle	John Travolta	1

- Above table,
  - Primary key associated to each dog that is related to each customer
  - One to many relationship
  - If more than one customer has a relationship with other data entries in other tables, defined to be a many to many relationship

Customer ID (PK)	Name	Email
1	John Travolta	john@travolta.com
2	Amy Cruise	amy@cruise.com
3	Amy Cruise's Husband	amy@cruiseHusband.com

ZIPCAR EXAMPLE

- Users can hire unique cars over a certain, non-permanent period of time.
- Foreign key updating cant work because erasing data is removes rent history.

Customer data - Table 1

Customer ID (PK)	Name	Email
1	John Travolta	john@travolta.com
2	Amy Cruise	amy@cruise.com
3	Amy Cruise's Husband	amy@cruiseHusband.com

Vehicle Data - Table 2

Vehicle ID (PK)	Name
1	Hummer
2	Carrola
3	Pruis

Many to Many Table - "Join Table"

Customer ID (FK)	Vehicle ID (FK)
1	2
2	1
3	3
2	1
2	3
1	2
3	2

MERGING DATA SETS TO QUERY DIFFERENT DATA

- In the case where data is related with each other in various ways, data sets needs to be joined to query many to many relationships
- Join foreign key to primary key
  - NULL data sets will be pulled and conjoined with

1. JOINING TABLES TOGETHER

1.1. Join two tables together for a specified data type

1.1.1. SELECT table1.column1, table2.column1, table2.column2
FROM table1
INNER JOIN table2
ON table1.columnName = table2.columnName
WHERE table2.name = dataType

1.1.1.1. Information available from various tables as compiler executes INNER JOIN first.

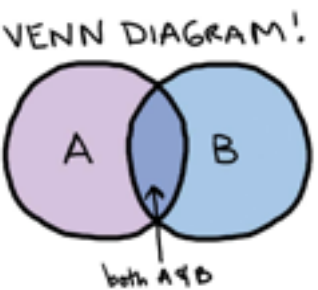
1.1.1.2. SELECT is the last function to execute

1.2. Join tables using information from a JOIN TABLE

1.2.1. SELECT * FROM table1
INNER JOIN joinTable
ON table1.columnID = table 2.columnID
INNER JOIN table3
ON table2.columnID = table3.columnID

WHERE table1.columnID = dataTYPE;

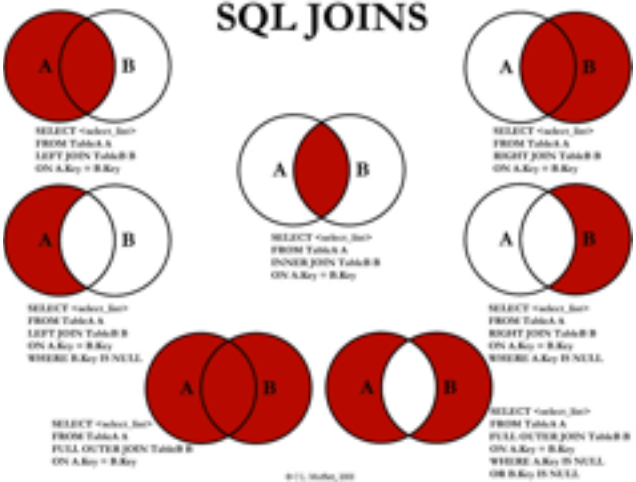
NOTE: '=' is strict equality and thus wont switching arguments on ON doesnt make a difference WHERE A = table1 and B is table2 and AB is INNER JOIN



1.2.2. Order the information according to various criteria and thus dataType

```
SELECT * FROM table1
INNER JOIN joinTable
  ON table1.columnID = table 2.columnID
INNER JOIN table3
  ON table2.columnID = table3.columnID

ORDER BY table1.columnName, table2.columnName
```



Examples - WORLD.sql

1. Find all languages spoken in Indonesia
   1.1. SELECT * FROM CountryLanguage
   INNER JOIN Country
   ON Country.Code = CountryLanguage.CountryCode
   WHERE Country.Name = 'Indonesia'

2. See a list of North American countries and their accompanying languages
   2.1. SELECT * FROM CountryLanguage
   INNER JOIN Country
   ON Country.Code = CountryLanguage.CountryCode
   WHERE Country.Name = 'Indonesia'
3. See a list of cities on China
   3.1.
4. See unique languages spoken in all Federal Republics
   SELECT DISTINCT CountryLanguage.Language FROM CountryLanguage
   INNER JOIN Country
   ON Country.Code = CountryLanguage.CountryCode

   WHERE Country.GovernmentForm = 'Federal Republic';

ERD - Entity related Diagram
- Draw.io
- Define column variableTypes for each table
- Google - erd app

SQL ZOO
- sqlzoo.net

Postgres - Database server: Postgress.app