My SQL

HOSTNAMES & WILDCARDS

- CREATE USER ‘bill’@’%.3sn.net’

  IDENTIFIED BY ‘secret!23x’;

  The @ sign is not in quotation marks. Usernames may contain wildcard characters such as %.

CREATE TABLE STATEMENT

- SELECT table_name, engine FROM information_schema.tables WHERE table_schema = ‘scratch’;

  Each table in a database is associated with a storage engine. This line of code gives us a feedback of table_names along with what kind of engine that they are using.

- -- create table with engine

  ```
  USE scratch;
  DROP TABLE IF EXISTS test;
  CREATE TABLE IF NOT EXISTS test (
     id INT AUTO_INCREMENT PRIMARY KEY,
     cname VARCHAR(128),
     localname VARCHAR(128)
  ) ENGINE = InnoDB;
  INSERT INTO test (cname, localname) SELECT name, localname FROM
  world.country;
  SELECT COUNT(*) FROM test;
  SELECT * FROM test;
  ```

SELECT table_name, engine FROM information_schema.tables WHERE table_schema = 'scratch';

DROP TABLE IF EXISTS test;

- Uses the scratch database, drops the table if it exists, and creates the table if it does not exist as a test. Test has 3 columns - id, cname, and localname. We also create the engine as InnoDB by placing the equal sign.

  We then insert some rows into the table. We pull the rows from the world database country table.

INNODB ENGINE

- The default storage engine for versions of MySQL
- The InnoDB engine is fully acid compliant - Atomicity, Consistency, Isolation, Durability.
- The database remains in a coherent and usable state.
- Row-level locking allows multiple read & write options to occur simultaneously on different rows in the same table without impacting database integrity.
- Foreign Key Constraints - ensure that updates do not result with inconsistencies.
- MVCC - Multi- Version Concurrency Control keeps info about old versions of changed rows in order to provide integrity. Allows queries to proceed.

MyISAM

- Was the default engine up to 2010. ISAM stand for Indexed Sequential Access Method
- Does not support transactions or foreign keys. It's only locking mechanism locks entire tables at a time.

THE MEMORY ENGINE

- Stores its contents in temporary memory.
- Vulnerable to crashes, hardware issues, or power outages.

SQL CLAUSES

- SELECT * FROM table;
- The semicolon is always allowed, beginning with a keyword. Technically the semicolon is a statement terminator, but in SQL it is more of a separator.
- SQL statements are not case sensitive.
- Comment example - SELECT * FROM Album; -- this is a comment
- /*

  This is a multi line comment

  /*
- # not a standard SQL comment

DATABASE ORGANIZATION

- My SQL is a relational database comprised of two dimensional tables comprised of rows & columns
- Field and column are interchangeable.
- The primary key will always be in databases. There will always be a unique key that is used to create relationships between tables.

SELECTING ROWS

- USE World;
- SELECT Name, LifeExpectancy AS ‘Life Expectancy’ FROM Country ORDER BY Name;
- Order in Alphabetical order by name, gives Life Expectancy a new column name.

SELECTING COLUMNS

- USE World;
- SELECT * FROM Country ORDER By Code;
- SELECT Name AS Country, Code as ISO, Region, Population as Pop From Country ORDER by Pop;
- Simply selects certain columns with the World dataset, and renames them accordingly, orders by Population size in ascending order (default).

COUNTING ROWS

- USE world;
- SELECT COUNT(*) FROM Country WHERE Population > 100000000 AND Continent = ‘Europe’;
- Returns the row count of the country.

INSERTING DATA

- USE scratch;
- SELECT * FROM Customer;
- INSERT INTO customer (name, address, city, state, zip) VALUES (' Fred Flinstone' , '123 Cobblestone Way' , 'Bedrock' , 'CA' , '91234');
- SELECT * FROM Customer;

UPDATING DATA

- USE scratch;
- SELECT * FROM Customer
- SELECT * FROM Customer WHERE name LIKE 'Jimi%';
- This code select everything that begins with Jimi, and % allows for anything that follows.
- SET SQL_SAFE_UPDATES = 0;

  UPDATE customer SET address = '123 Music Avenue' , zip = '98056' WHERE name LIKE 'Jimi%';

  SELECT * FROM customer WHERE name LIKE 'Jimi%';
- The first line allows you to update a row. It is a safety feature. The code allows you to update specific columns where the classifications meet with the first name Jimi.

DELETING DATA

- USE scratch;
- CREATE TABLE test (a INT, b VARCHAR(16), c VARCHAR(16));
- INSERT INTO test VALUES (1, 'this', 'right here!');
- INSERT INTO test VALUES (2, 'that', 'over there!');
- INSERT INTO test VALUES (3, 'another', 'nowhere.');
- INSERT INTO test VALUES (4, 'again' , 'guess where?');
- INSERT INTO test VALUES (1, 'one more' , 'everywhere!');
- DELETE FROM test WHERE a = 2;
- SELECT * FROM test;
- DROP TABLE test;

- SELECT * FROM customer WHERE name LIKE 'Jimi%' OR name LIKE 'Fred%';
- DELETE FROM customer WHERE name LIKE 'Jimi%' OR name LIKE 'Fred%';
- SELECT * FROM customer;
- Simply selects and deletes items rows with certain keywords.

JOINING QUERIES ACROSS TABLES

- SELECT * FROM track;

  SELECT * FROM album;
- SELECT a.artist AS Artist, a.title AS Album, t.track_number AS 'Track Num',

    t.title AS Track, t.duration AS Seconds

   FROM album AS a

   JOIN track AS t ON a.id = t.album_id

   ORDER BY a.artist, a.title, t.track_number;
- Where the album ID in the track table intersects with the id column in the album table is where the results get joined.
- The top code renames the columns. The bottom code provides order.
- This is what makes relationships useful in MySQL databases.

FINDING DATABASES, TABLES AND COLUMNS

- SHOW databases;
- USE scratch;

  SHOW tables;
- DESCRIBE customer;

This will give a list of columns within the table, their type, whether or not they allow null, if it is a primary key.

DATA TYPES- WHAT ARE DATA TYPES?

- Fundamental - Numeric, String, Data & Time, Specialty.
- Integer types - 47 , 1000000047
- Fixed Point Types - 3.47, 1000000.047
- Floating Point Types - 3.47, 100.47e12   used for apps where accuracy is not as important
- Fixed length strings - 90210, SW1A, 2AA   postal codes, padded with spaces for storage, stripped when retrieved
- Variable length strings - Subterranean Homesick Blues
- Binary Strings - Not treated as text, with no character set. BINARY (8)
- etc.

THE CREATE TABLE STATEMENT

- CREATE TABLE test (

```
   id INT,
   name VARCHAR(255),
   address VARCHAR(255),
   city VARCHAR(255),
   state CHAR(2),
   zip CHAR(10)
);
```

- The varchar has a maximum length of 255 variable length character string. The state, and zip are fixed length character strings.
- DESCRIBE test; shows the field, type, null, key
- SHOW TABLE STATUS; tells what engine is being used, row format, rows, average row length, data length.

NUMERIC TYPES

- Integer Type
- Decimal Type (p, s) precision, scale Example (9, 2) 1234567.89
- Float Type - $13.500001 never use a float for money. It is too accurate, you want a decimal number

DATE & TIME TYPES

- USE scratch;
- SELECT NOW(); - will show current date & time in SQL format.
- SHOW VARIABLES LIKE ‘%timezone’; -
- SET time_zone = ‘+00:00’; England Time

  SELECT NOW( );
- SET time_zone = 'SYSTEM'; Wherever you are located - Mountain Time example
- SELECT UTC_TIMESTAMP( );
- DROP TABLE IF EXISTS temp;

  ```
  CREATE TABLE temp (
    id INT UNSIGNED UNIQUE AUTO_INCREMENT PRIMARY KEY,
  ```

```
  stamp TIMESTAMP,
  name VARCHAR(64)
);
INSERT INTO temp (name) VALUES ('this');
INSERT INTO temp (name) VALUES ('that');
INSERT INTO temp (name) VALUES ('other');
SELECT * FROM temp;
```

- The code creates a table with 3 variables. ID, Stamp, & name.
- stamp should become - stamp DATETIME DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP
- This will allow the timestamp in the database.

STRING TYPES

- Character Strings, CHAR (length)   ex: CHAR(5) Always uses 5 bytes
- VARCHAR(25)   ex: The string will have up to 25 characters. Uses up to 25 + 1 bytes.
- Binary Strings,

ENUMERATION TYPES

- USE scratch;
- DROP TABLE IF EXISTS test;

```
CREATE TABLE test (
  id INT UNSIGNED UNIQUE AUTO_INCREMENT PRIMARY KEY,
  a ENUM( 'Pablo', 'Henri', 'Jackson' )
);

INSERT INTO test ( a ) VALUES ( 'Pablo' );

INSERT INTO test ( a ) VALUES ( 'Henri' );

INSERT INTO test ( a ) VALUES ( 'Jackson' );

INSERT INTO test ( a ) VALUES ( 'Pablo,Jackson,Henri,Henri,Henri' );

INSERT INTO test ( a ) VALUES ( 1 );

INSERT INTO test ( a ) VALUES ( 2 );

INSERT INTO test ( a ) VALUES ( 3 );

INSERT INTO test ( a ) VALUES ( 4 );

INSERT INTO test ( a ) VALUES ( 5 );

INSERT INTO test ( a ) VALUES ( 6 );

INSERT INTO test ( a ) VALUES ( 7 );

SELECT * FROM test;
```

THE SERIAL TYPE ALIAS

- CREATE TABLE test (

  ```
          id SERIAL,

          a VARCHAR(32)

  );
  ```

- The id SERIAL is an alias for big int, unsigned, not null, unique, primary key.
- The downside is the big int takes up more bytes.

COMPARISON OPERATORS

- SELECT O = 1;
- The outcome is 0 meaning it is false. 1 would represent true.

LOGICAL OPERATORS

- SELECT 1 AND 1;
- Will get a true result. True and True is True.
- SELECT 1 OR 0;   True or False is True. Only 1 of the operands needs to be true.
- SELECT 1 XOR 0;   XOR REQUIRES THAT ONE SIDE OF THE OPERATIONS IS TRUE AND THE OTHER IS FALSE.
- SELECT 1 IS NOT TRUE;   Will come out false.
- SELECT 9 IN (1, 5, 9)   come out true.
- USE world;

  ```
  SELECT Name AS 'Country', ROUND(Population / 1000000) AS 'PopMM'
    FROM Country
    WHERE Population > 50000000 AND Continent IN ('Asia', 'Europe')
    ORDER BY Population DESC;
  ```
- Selecting Name category & renaming as country. We are taking that category and dividing by 1,000,000 then rounding to nearest whole number. That category then gets relabeled as ‘PopMM’.
- Now we call on it as the renamed variable FROM Country we are selecting data where the population is greater than 50,000,000 AND continent category is in ‘Asia’ or ‘Europe’. Finally we want the numbers to be descending.

- USE album;

  SELECT t.title AS 'Track', t.track_number AS 'Track No', a.title AS 'Album',

    a.artist AS 'Artist', t.duration AS 'Seconds'

   FROM Album AS a

   JOIN Track AS t ON t.album_id = a.id

   WHERE t.duration > 120 AND t.track_number > 3

   ORDER BY t.duration DESC;

ARITHMETIC OPERATORS

- SELECT 5 + 3;   8
- SELECT 5 MOD 3;   Remainder of 2.
- SELECT 5 / 0;   division by 0 results in a null result

OPERATOR PRECENDENCE

- SELECT 5 + 6 *7 - 8;   39
- SQL follows order of operations

THE CASE STATEMENT

- USE scratch;

  DROP TABLE IF EXISTS booltest;

  CREATE TABLE booltest (a INTEGER, b INTEGER);

  INSERT INTO booltest VALUES (1, 0);

  SELECT * FROM booltest;

- Use the scratch database, drop the table if it exists booltest. Then create the table booltest with two rows: an Integer, and an Integer. Then insert into the booltest values of 1 & 0. Then we select everything from booltest table.
- SELECT

  CASE WHEN A < 5 THEN ‘true’ ELSE ‘false’ END AS boolA,

  CASE WHEN b> 0 THEN ‘true’ ELSE ‘false’ END AS boolB

  FROM booltest

  ;
- Returns true & false as a string in a table.

THE IF FUNCTION

- USE scratch;

  DROP TABLE IF EXISTS booltest;

  CREATE TABLE booltest (a INTEGER, b INTEGER);

  INSERT INTO booltest VALUES (1, 0);

  SELECT * FROM booltest;
- SELECT IF(a < 5, ‘true’, ‘false’) FROM booltest;
- The result is the string value of true because a is less than 5.

STRING FUNCTIONS

- USE world;
- SELECT name FROM country WHERE name LIKE ‘_a%’ ORDER BY name;
- This code returns all countries where ‘a’ is in the second position of the name.

- The underscore is used as a single character wildcard.
- SELECT name FROM country WHERE STRCMP(name, ‘France’) <= 0 ORDER BY name;
- This code provides names of countries that sort lower than France.

REGULAR EXPRESSIONS

- USE world;

  SELECT Name FROM country WHERE Name RLIKE ‘y$’ ORDER BY Name;
- Selects all names from the category country where the name ends in the letter y.
- SELECT Name FROM country WHERE Name RLIKE ‘[xy][ai]’ ORDER BY Name;
- This gets countries that have an x or a y, which is then followed by an a or an i.

STRING CONCATENATION

- SELECT CONCAT(‘This’, ‘and’ , ‘that’);
- Output is This and that.

NUMERIC CONVERSIONS

- SELECT HEX(32742);   OUTPUT IS 7FE6.
- SELECT OCT(32742);   OUTPUT IS 7746
- SELECT BIN(32742);   OUTPUT IS 111111111100110
- SELECT CONV(32742, 10, 16);   OUTPUT IS 7FE6 from base 10 to base 16

TRIMMING AND PADDING

- USE scratch;

  SELECT * FROM customer WHERE name LIKE TRIM ‘ Bill Smith ‘;
- Because there are spaces on both sides of the name there are no results. However when the function trim is used you receive the right record. This applies for leading and trailing.
- SELECT CONCAT(‘:’ , TRIM(‘   Bill Smith   ‘),   ‘:’);
- The trim function takes away the spaces but leaves the colons next to the name.
- SELECT CONCAT(‘:’ , LTRIM(‘   Bill Smith   ‘),   ‘:’);
- LTRIM takes away only the spaces on the left hand side. Same for RTRIM.
- SELECT CONCAT(‘:’ , TRIM(‘x’ FROM ‘ xxxxx   Bill Smith xxxxx   ‘),   ‘:’);
- This code trims the x’s specifically.
- SELECT LPAD(‘Price’, 20, ‘.’);
- This pads periods in the left hand side of the string until it reaches 20 character including the string.

CASE CONVERSION

- USE scratch;
- SELECT UPPER(name) FROM customer;   provides names in all upper case.
- SELECT LOWER(name) FROM customer;   provides names in all lower case.
- SELECT CONCAT(UPPER(SUBSTRING(name, 1, 1)), LOWER(SUBSTRING(name, 2))) FROM customer;

- The code capitalizes the first character at the first position. The other letters are all lowercase starting at the second position.

SUBSTRINGS

- SELECT SUBSTRING(‘this is a string’, 6)
- The index does not start at 0. Output - ‘is a string’
- SELECT SUBSTR is an alias for substring.
- SELECT SUBSTR(‘this is a string’, 6, 4); provides the first 4 characters after position 6.
- SELECT SUBSTR(‘this is a string’, -6); gives the character starting with 6 from the end.
- SELECT SUBSTRING_INDEX(‘this is a string’, ‘ ‘, 1); will return all of the characters up to the first delimiter. In this case it is a space so it returns ‘this’. You can change it to the second delimiter by changing to a 2.

VALUES FUNCTIONS

- SELECT ABS(-47.36); gives the absolute value = 47.36
- SELECT CEILING(12.2); gives the result of 13. It is the lowest value that is not lower than the operand.
- SELECT FLOOR(12.9); 12
- SELECT ROUND(17.4) 17
- SELECT TRUNCATE(42.973, 2); 42.97.
- SELECT TRUNCATE(99942.973, -2); The code truncates to the left of the decimal

BASIC MATH

SELECT PI( ) + 0.0000000000000;	returns pi to the extent of the decimals.
SELECT POWER(8, 2);	64
SELECT SQRT(64);	8
SELECT POWER(4096, ¼);	provides the 4th root of 4096 = 4
SELECT RAND();	gives a random number
SELECT RAND(42);	gives it a seed.

SIMPLE TRIGONOMETRY

- SELECT SIN(2);
- SELECT ASIN(.2);
- SELECT COS(PI());
- SELECT ACOS(.5);
- SELECT TAN(PI);
- SELECT ATAN(PI);
- SELECT COT(2);

LOGARITHMS

SELECT LN(2);	gives us the natural logarithm of 2. = .693147
SELECT LOG(10, 100);	returns the log of 100 to the base of the first argument.
SELECT LOG10(100);	returns the same result.
SELECT EXP(1);	The inverse of log. 2.71

RADIANS & DEGREES

- SELECT DEGREES(PI()); OUTPUT IS 180 degrees.
- SELECT RADIANS(180); OUTPUT IS 3.14

STRING CONCATENATION

- SELECT ‘string1’ || ‘string2’;
- This is used as a logical or function. Returns 0 because neither evaluates to true.
- SELECT TRUE || FALSE;
- Outcome is 1 because true is true.
- SELECT CONCAT(‘string1’, ‘string2); outcome is string1string2

QUOTE MARKS

- SELECT ‘this is a string’;
- USE scratch;

  SELECT “name”, “address” FROM customer;
- The result selects all the literal strings form name, and address in each row. The double quotes are literal strings. The solution to this is use backticks.
- SELECT \`name\`, \`address\` FROM customer;

INTEGER DIVISION;

- SELECT 47 / 3; 15.6667
- SELECT DIV(47, 3); Results with an error in syntax.

- SELECT 47 DIV 3;   div is an operator is SQL. Feedsback integer division of 15 w/ no remainder.