SQL NOTES

Page 15

You may be wondering what “views” are. Do not worry about them for now. They are basically prebuilt SQL queries that are used so frequently, they are conveniently stored in the database.

Page 22

You do not have to pull all columns in a SELECT statement. You can also pick and choose only the columns you are interested in. The following query will only pull the CUSTOMER_ID and NAME columns:

SELECT CUSTOMER_ID, NAME FROM CUSTOMER;

A single SQL statement can optionally end with a semicolon (;), as shown in the previous examples. However, the semicolon is neces- sary to run multiple SQL statements at once,

Page 23

Suppose we wanted to generate a calculated column called TAXED_PRICE that is 7% higher than PRICE. We could use a SELECT query to dynamically calculate this for us

SELECT PRODUCT_ID, DESCRIPTION, PRICE, PRICE * 1.07 AS TAXED_PRICE FROM PRODUCT;

Page 25

Every database platform has built-in

Page 26

functions to assist with these kinds of operations, and SQLite provides a round() function that accepts two arguments in parentheses separated by a comma: the num- ber to be rounded, and the number of decimal places to round to.

SELECT PRODUCT_ID, DESCRIPTION, PRICE, round(PRICE * 1.07, 2) AS TAXED_PRICE FROM PRODUCT;

Page 27

oncatenation, which merges two or more pieces of data together. The concatenate operator is specified by a double pipe (||), and you put the data values to concatenate on both sides of it.

For instance, you can concatenate the CITY and STATE fields from the CUSTOMER table as well as put a comma and space between them to create a LOCATION value

SELECT NAME, CITY || ', ' || STATE AS LOCATION FROM CUSTOMER;

Page 31

We can also qualify inclusive ranges using a BETWEEN statement

SELECT * FROM station_data WHERE year BETWEEN 2005 and 2010

AND

SELECT * FROM station_data WHERE year >= 2005 AND year <= 2010

Page 32

SELECT * FROM station_data WHERE MONTH = 3 ORMONTH=6 ORMONTH=9 OR MONTH = 12

This is a little verbose. A more efficient way to accomplish this is using an IN statement to provide valid list of values:

SELECT * FROM station_data WHERE MONTH IN (3,6,9,12)

You can use other math expressions in your WHERE statements too.

In English, this means “give me all months where dividing by 3 gives me a remainder of 0”: SELECT * FROM station_data WHEREMONTH%3=0