**INDEX** (BTree data structure)

Indexes are used to find values within a specific column more quickly

MySQL normally searches sequentially through a column

The longer the column, the more expensive the operation is

UPDATE takes more time, SELECT takes less time -- (index is for much easier locating)

>

**CREATE INDEX** last name idx -- (creating an index)

ON customers(last\_name);

**SHOW INDEXES FROM** customers; -- (showing in index)

>

CREATE INDEX last name first name idx

ON customers(last\_name, first \_ name); (multi column indexes)

SHOW INDEXES FROM customers;

>

ALTER TABLE customers

**DROP INDEX** last name idx; -- (drop index)

-**subquery**

a query within another query

query(subquery) (completing the sub query first)

>

SELECT first\_name, last\_name, hourly\_pay,

(SELECT AVG(hour1y\_pay) FROM employees) AS avg\_pay

FROM employees;

>

SELECT first\_name, last\_name, hourly\_pay (greater than avg hourly pay)

FROM employees

WHERE hourly\_pay > (SELECT AVG(hourly\_pay) FROM employees);

>

**SELECT DISTINCT** customer\_id -- (distinct gets rid of repeats)

FROM transactions

WHERE customer\_id IS NOT NULL;

>

SELECT first\_name, last\_name

FROM customers

WHERE customer id IN (NOT IN opposite of IN)

(SELECT DISTINCT customer\_id

FROM transactions

WHERE customer\_id IS NOT NULL);

**GROUP BY** = aggregate all rows by a specific column

often used with aggregate functions

ex. SUM(), MAX(), MIN(), AVG(), COUNT()

>

SELECT SUM(amount), order\_date

FROM transactions

GROUP BY order\_date; (getting the sum of amounts by date)

>

SELECT COUNT(amount), customer\_id

FROM transactions

GROUP BY customer\_id (customers visiting more than once)

**HAVING COUNT**(amount) > 1; (HAVING instead of where) (works the same)

**ROLLUP**, extension of the GROUP BY clause

produces another row and shows the GRAND TOTAL (super-aggregate value)

SELECT SUM (amount), order\_date

FROM transactions

GROUP BY order date WITH ROLLUP; (grand total of the amount)

**ON DELETE SET NULL** = When a FK is deleted, replace FK with NULL

**ON DELETE CASCADE** = Wen a FK is deleted, delete row (Foreign Key)