

SQL - Bootcamp

CS102 - April 29, 2020





Review - Important Queries

SELECT...FROM...WHERE

- Fundamental structure of all SQL queries

GROUP BY...HAVING

- Remember: HAVING applies to groups **only**. WHERE applies to ungrouped data.

LIMIT

- Limits the number of results returned

ORDER BY

- Used to sort results based on a specific column. Can specify ASC (default) or DESC

SELECT DISTINCT

- Eliminates duplicates in results



Review - Important Queries

SUM, COUNT, MIN, MAX, AVG

- Aggregation functions

LIKE

- s **LIKE** p: pattern matching on strings
- p may contain two special symbols:
 - % = any sequence of characters
 - _ = any single character
- Use LIKE as part of WHERE clause (e.g., SELECT __ FROM __ WHERE s LIKE '%substring%')



Review - Important Queries

JOIN

- Combines data from two tables based on a specified shared column, before completing query and returning results.
- Usually, this functionality can also be achieved with an extra statement in the WHERE clause
- Example (source: [CS145](#))

```
Product(PName, Price, Category, Manufacturer)
Company(CName, StockPrice, Country)
```

Several equivalent ways to write a basic join in SQL:

```
SELECT PName, Price
FROM   Product, Company
WHERE  Manufacturer = CName
      AND Country='Japan'
      AND Price <= 200
```

```
SELECT PName, Price
FROM   Product
JOIN   Company
ON     Manufacturer = Cname
WHERE  Price <= 200
      AND Country='Japan'
```



Types of JOINS

- INNER JOIN (default) - excludes NULL values
- OUTER JOIN
 - FULL
 - LEFT - will return (a, NULL)
 - RIGHT - will return (NULL, b)

SQL CHEAT SHEET <http://www.sqltutorial.org>



QUERYING DATA FROM A TABLE

SELECT c1, c2 FROM t;
Query data in columns c1, c2 from a table

SELECT * FROM t;
Query all rows and columns from a table

**SELECT c1, c2 FROM t
WHERE condition;**
Query data and filter rows with a condition

**SELECT DISTINCT c1 FROM t
WHERE condition;**
Query distinct rows from a table

**SELECT c1, c2 FROM t
ORDER BY c1 ASC (DESC);**
Sort the result set in ascending or descending order

**SELECT c1, c2 FROM t
ORDER BY c1
LIMIT n OFFSET offset;**
Skip *offset* of rows and return the next *n* rows

**SELECT c1, aggregate(c2)
FROM t
GROUP BY c1;**
Group rows using an aggregate function

**SELECT c1, aggregate(c2)
FROM t
GROUP BY c1
HAVING condition;**
Filter groups using HAVING clause

QUERYING FROM MULTIPLE TABLES

**SELECT c1, c2
FROM t1
INNER JOIN t2 ON condition;**
Inner join t1 and t2

**SELECT c1, c2
FROM t1
LEFT JOIN t2 ON condition;**
Left join t1 and t2

**SELECT c1, c2
FROM t1
RIGHT JOIN t2 ON condition;**
Right join t1 and t2

**SELECT c1, c2
FROM t1
FULL OUTER JOIN t2 ON condition;**
Perform full outer join

**SELECT c1, c2
FROM t1
CROSS JOIN t2;**
Produce a Cartesian product of rows in tables

**SELECT c1, c2
FROM t1, t2**
Another way to perform cross join

**SELECT c1, c2
FROM t1 A
INNER JOIN t2 B ON condition;**
Join t1 to itself using INNER JOIN clause

USING SQL OPERATORS

**SELECT c1, c2 FROM t1
UNION (ALL)
SELECT c1, c2 FROM t2;**
Combine rows from two queries

**SELECT c1, c2 FROM t1
INTERSECT
SELECT c1, c2 FROM t2;**
Return the intersection of two queries

**SELECT c1, c2 FROM t1
MINUS
SELECT c1, c2 FROM t2;**
Subtract a result set from another result set

**SELECT c1, c2 FROM t1
WHERE c1 [NOT] LIKE pattern;**
Query rows using pattern matching %, _

**SELECT c1, c2 FROM t
WHERE c1 [NOT] IN value_list;**
Query rows in a list

**SELECT c1, c2 FROM t
WHERE c1 BETWEEN low AND high;**
Query rows between two values

**SELECT c1, c2 FROM t
WHERE c1 IS [NOT] NULL;**
Check if values in a table is NULL or not