

# SQL In 10 Minutes

**Summarizing & Manipulating Data - Vibhu Sapra** 

# Manipulate & Sum

# Overview

- Extract the specific data you want
  - Not very portable
  - Each DBMS uses different syntax
  - ▶ Text, Date, and Numeric Manipulation
- Summarize data
  - Actually pretty consistent across DBMS
  - Avg, Count, Max, Min, Sum, etc.



Inconsistency 🤦





# **Data Manipulation**

- Manipulate text / dates / numbers
- Every DBMS uses different syntax
- Can still accomplish the same tasks, just different code
  - Text Functions
  - Numeric Functions
  - Date & Time Functions
  - Formatting Functions
  - System Functions



# **Data Manipulation**

Every DBMS uses different syntax

Extract part of a string

DB2, Oracle, PostgreSQL, and SQLite use SUBSTRING(). MariaDB, MySQL, and SQL Server use SUBSTRING().

Oracle uses multiple functions, one for each conversion type.

DB2, PostgreSQL, and SQL Server use CAST(). MariaDB,

MySQL, and SQL Server use CONVERT().

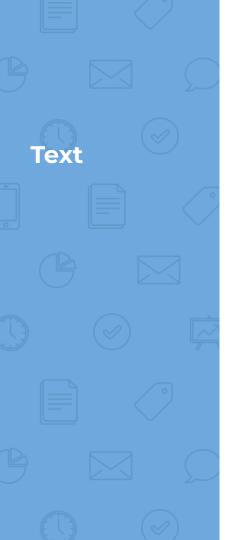
Get DB2 and PostgreSQL use CURRENT\_DATE. MariaDB and MySQL current use CURDATE(). Oracle uses SYSDATE. SQL Server uses date GETDATE(). SQLite uses DATE().





# **Text Manipulation**

- Saw RTRIM() last time
- Not case sensitive
  - upper(), UPPER(), Upper(), substr(), SUBSTR(), SubStr() all work the same
- Many more text manipulation functions
- Let's start with Upper() -> Uppercase text



# **Text Manipulation**

SELECT vend\_name, UPPER(vend\_name) AS vend\_name\_upcase
FROM Vendors
ORDER BY vend name;

### Output \*

### Click here to view code image

vend\_name

Bear Emporium
Bears R Us
Doll House Inc.
Fun and Games
Furball Inc.
Jouets et ours

vend\_name\_upcase

BEAR EMPORIUM
BEARS R US
DOLL HOUSE INC.
FUN AND GAMES
FURBALL INC.
JOUETS ET OURS



# **Common Functions**

### **Function**

### Description

LEFT() (or use substring function LENGTH() (also DATALENGTH() or

LEFT() (or use substring function Returns characters from left of string

LEN())

LOWER ()

LTRIM()

RIGHT() (or use substring function)

RTRIM()

SUBSTR() or SUBSTRING()

SOUNDEX ()

UPPER()

Returns the length of a string

Converts string to lowercase

Trims white space from left of string

Returns characters from right of string

Trims white space from right of string

Extracts part of a string (as noted in Table 8.1)

Returns a string's SOUNDEX value

Converts string to uppercase



# Soundex Fn

- Fix typos in names and stuff
- Convert text to alphanumeric pattern
  - Not in PostgreSQL
  - Not default in SQLite
- Customer named "Michelle Green"
  - Typo and should be "Michael Green"

# **Soundex**

# Soundex Fn

```
SELECT cust_name, cust_contact
FROM Customers
WHERE cust contact = 'Michael Green';
```

# Output \*

# Click here to view code image

```
cust_name cust_contact
```



# Soundex Fn

### Click here to view code image

```
SELECT cust_name, cust_contact
FROM Customers
WHERE SOUNDEX(cust_contact) = SOUNDEX('Michael Green');
```

# Output \*

### Click here to view code image

```
cust_name
-----
Kids Place
```

```
cust_contact
-----
Michelle Green
```





# **Date & Time**

- Values sorted in special format so they can be indexed quickly
  - ▶ Internal format = unreadable
- Most inconsistent between DBMS :(
- Basic querying

### Click here to view code image

```
SELECT order_num
FROM Orders
WHERE DATEPART(yy, order_date) = 2020;
```

### Output \*



# **Date & Time**

- Values sorted in special format so they can be indexed quickly
  - ▶ Internal format = unreadable
- Most inconsistent between DBMS :(
- Basic querying DATEPART ( )
  - Only for SQL Server
- Every DBMS is different...
  - Know the one you need

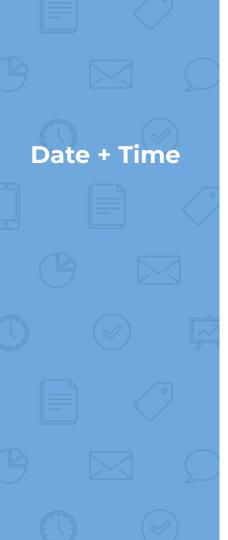


# **SQL Version**

### Click here to view code image

```
SELECT order_num
FROM Orders
WHERE DATEPART(yy, order_date) = 2020;
```

# Output \*

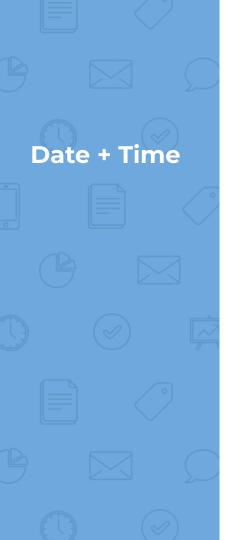


# **Oracle Version**

# Input •

# Click here to view code image

```
SELECT order_num
FROM Orders
WHERE EXTRACT (year FROM order_date) = 2020;
```



# **PostgreSQL Version**

# **Input** ▼

# Click here to view code image

```
SELECT order_num
FROM Orders
WHERE DATE_PART('year', order_date) = 2020;
```

PostgreSQL also can do datepart() and extract



# **Date & Time**

- DB2, MySQL, and MariaDB have all sorts of date manipulation functions, but not DATEPART()
  - Users can use a function named YEAR()
- SQLite is also different

SQLite is a little trickier:

### **Input** ▼

Click here to view code image

```
SELECT order_num
FROM Orders
WHERE strftime('%Y', order_date) = '2020';
```





# **Numeric Functions**

### Uniform and consistent!

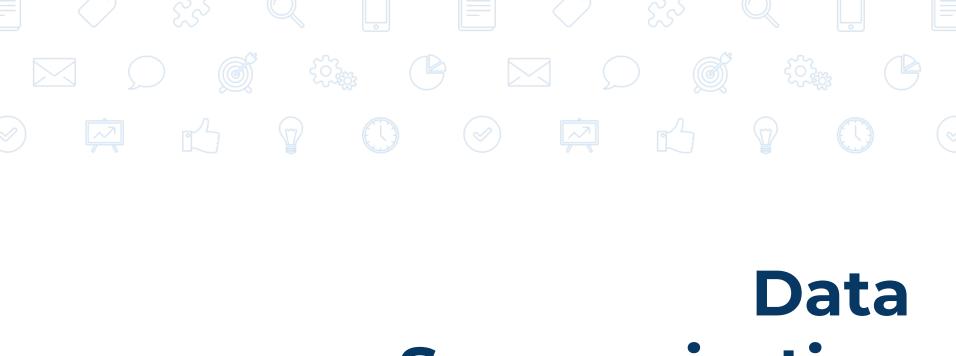
 Algebraic, Trig, Geometric, etc. transformations

<b>Function</b>	Description
ABS()	Returns a number's absolute value
COS()	Returns the trigonometric cosine of a specified angle
EXP()	Returns the exponential value of a specific number
PI()	Returns the value of PI
PI() SIN()	Returns the value of PI Returns the trigonometric sine of a specified angle



# Summary

- Very inconsistent
- Still good to use fast & scalable
- Relevant in the next section on Summarizing Data
- Questions break?



# Summarization

Consistency \*\*



# **Data Summarization**

- Summarize data w/o extracting it all
  - More efficient to pull what you need
  - SQL is fast
- Aggregation functions are pretty universal between DBMS
- Very Straightforward and "easy"



# Aggregation Fn ()

**Table 9.1** SQL Aggregate Functions

<b>Function</b>	Description
	~ c 5 c 1 - p c 1 c 1 .

AVG()	Returns a column's average value
COUNT ()	Returns the number of rows in a column
MAX()	Returns a column's highest value
MIN()	Returns a column's lowest value
SUM()	Returns the sum of a column's values



# Average Fn ()

Returns the average of all columns

### **Input** ▼

### Click here to view code image

SELECT AVG(prod\_price) AS avg\_price
FROM Products;

### Output \*

```
avg_price
------
6.823333
```

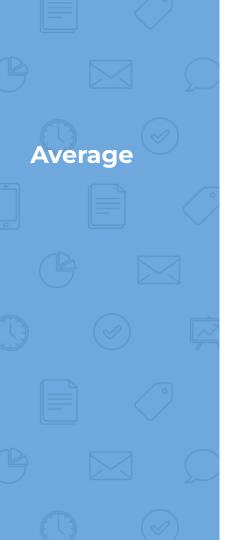
### Input \*

### Click here to view code image

```
SELECT AVG(prod_price) AS avg_price
FROM Products
WHERE vend_id = 'DLL01';
```

### Output ▼

```
avg_price
-----3.8650
```



# Average Fn ()

- Returns the average of all columns
- Can only do the average of one column at once
- Null Values = ignored



# Average Fn ()

Returns the average of all columns

### **Input** ▼

### Click here to view code image

SELECT AVG(prod\_price) AS avg\_price
FROM Products;

### Output \*

```
avg_price
------
6.823333
```

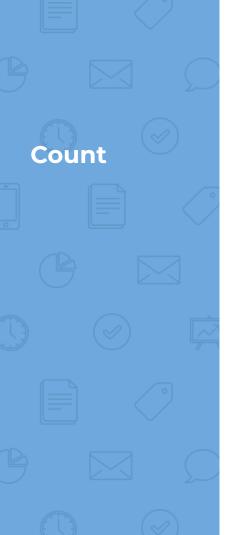
### Input \*

### Click here to view code image

```
SELECT AVG(prod_price) AS avg_price
FROM Products
WHERE vend_id = 'DLL01';
```

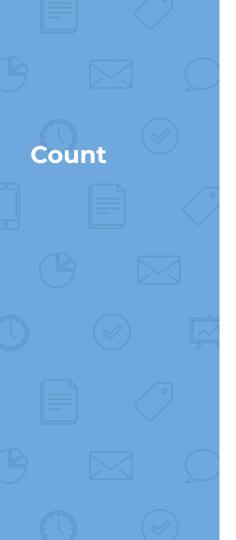
### Output ▼

```
avg_price
-----3.8650
```



# Count Fn ()

- Determine number of rows in a table that meet a specific criteria
- Count (\*) = number of all rows regardless of null
- Count (column) = number of rows that have a value in a column ignoring null



# Count Fn ()

- Count (\*) = total number of customers
- Count (cust\_emails) = total number of customers that gave their emails
  - Ignores null values



# Count Fn ()

# **Input ▼**

SELECT COUNT(\*) AS num\_cust
FROM Customers;

# Output \*

num\_cust

### **Input ▼**

Click here to view code image

SELECT COUNT(cust\_email) AS num\_cust FROM Customers;

### Output \*

num\_cust

3



# Min & Max Fn ()

- Returns the Max / Min values
- Ignore Null values
- \* Max can be used with non-numeric but not Min \*

SELECT MAX (prod\_price) AS max\_price FROM Products;

SELECT MIN(prod\_price) AS min\_price FROM Products;

# Output \*

max\_price

11.9900

Output \*

min\_price -----3.4900



# Sum Fn()

- Returns the sum (total) of all values
- Can be used to calc total (price \* total quantity (ignore null)

```
SELECT SUM(quantity) AS items_ordered FROM OrderItems
WHERE order num = 20005;
```

SELECT SUM(item\_price\*quantity) AS total\_price
FROM OrderItems
WHERE order\_num = 20005;

### Output \*

items\_ordered

200

### Output \*

total\_price -----1648.0000



# **General Tips**

- To perform calculations on all rows, specify the ALL argument
  - On by default
- To include only unique values, specify the **DISTINCT** argument.
  - Useful for things like average price of items

```
SELECT AVG (DISTINCT) prod_price) AS avg_price FROM Products
WHERE vend_id = 'DLL01';
```

### Output \*

# Tips

# Combine Fns ()

### Click here to view code image

```
SELECT COUNT(*) AS num_items,

MIN(prod_price) AS price_min,

MAX(prod_price) AS price_max,

AVG(prod_price) AS price_avg

FROM Products;
```

### Output \*

### Click here to view code image

num_items	price_min	price_max	<pre>price_avg</pre>
9	3.4900	11.9900	6.823333



# Questions?