Aggregating Data

Mihaela Danci
Data Analyst

linkedin.com/in/mihaela-danci/

Aggregate Functions

```
SELECT OrderDate, COUNT(OrderNumber) AS Orders
FROM Orders
WHERE SalesTeritorry = ‘USA’
GROUP BY OrderDate
HAVING COUNT(OrderNumber) > 1
ORDER BY OrderDate DESC;
```

Aggregate Functions

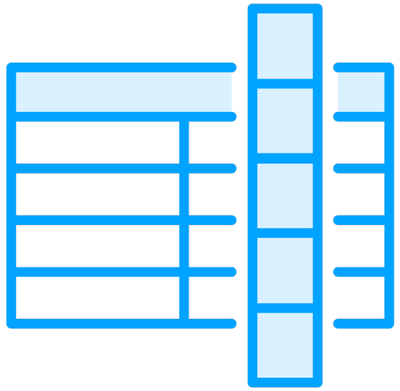
COUNT()

MIN()

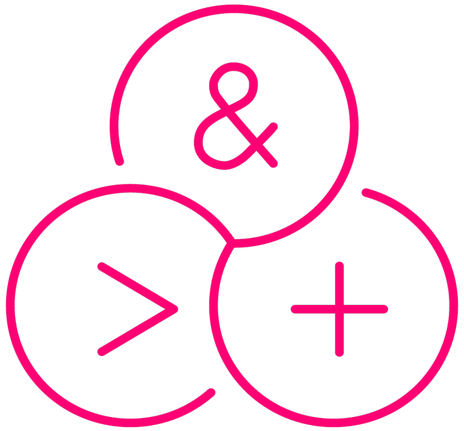
MAX()

AVG()

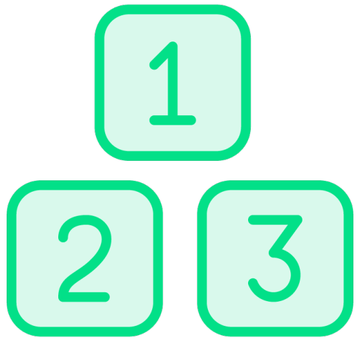
Aggregate Functions



Input



Operations



Result

Product	Stock
Bike	10
Jacket	20
Helmet	NULL
Shoes	20
T-shirt	30

Product	Stock
Bike	10
Jacket	20
Helmet	NULL
Shoes	20
T-shirt	30

```
SELECT COUNT(*)
FROM Product;

5
```

```
SELECT COUNT(Stock)
FROM Product;

4
```

Product	Stock
Bike	10
Jacket	20
Helmet	NULL
Shoes	20
T-shirt	30

```
SELECT COUNT(*)
FROM Product;

5
```

```
SELECT COUNT(Stock)
FROM Product;

4
```

Except for COUNT(*) which counts all rows, all the aggregate functions ignore NULLs.

Product	Stock
Bike	10
Jacket	20
Helmet	NULL
Shoes	20
T-shirt	30

```
SELECT SUM(Stock) / COUNT(*)
FROM Product;
16
```

```
SELECT AVG(Stock)
FROM Product;
 20
```

Distinct Values

```
SELECT COUNT(DISTINCT(Stock)
FROM Product;
 3
```

Product	Stock
Bike	10
Jacket	20
Helmet	NULL
Shoes	20
T-shirt	30

Distinct Values

```
SELECT COUNT(DISTINCT(Stock)
FROM Product;
 3
```

Product	Stock
Bike	10
Jacket	20
Helmet	NULL
Shoes	20
T-shirt	30

Distinct Values

```
SELECT SUM(DISTINCT(Stock)
FROM Product;
 60
```

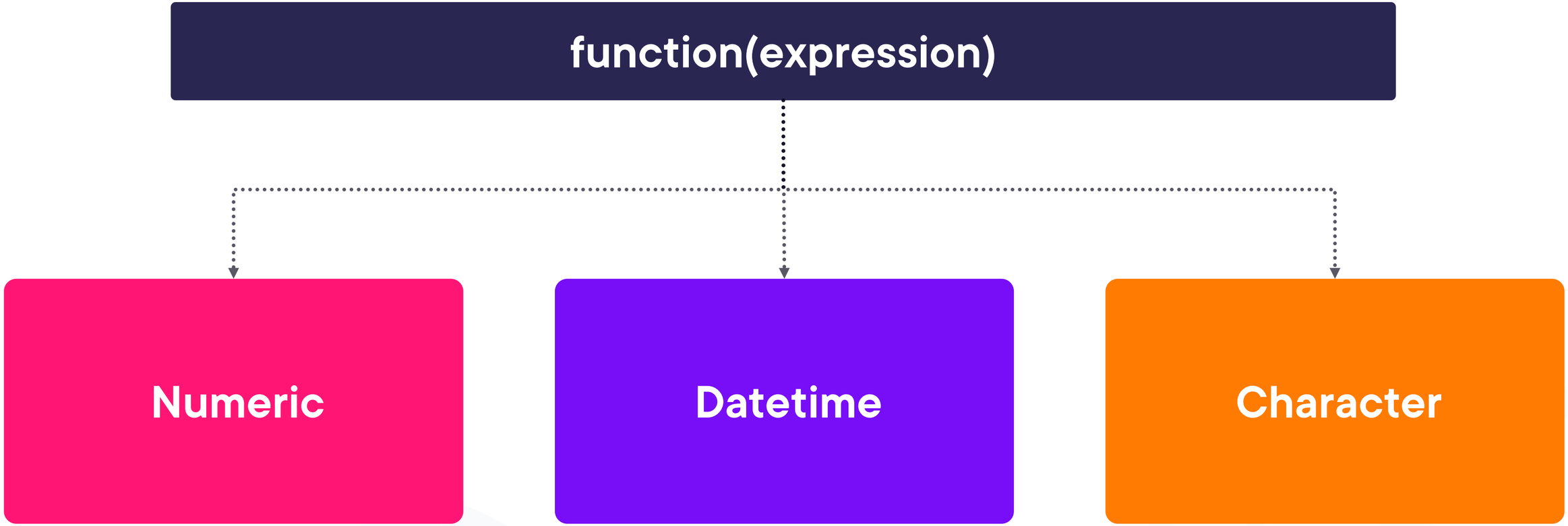
Product	Stock
Bike	10
Jacket	20
Helmet	NULL
Shoes	20
T-shirt	30

10

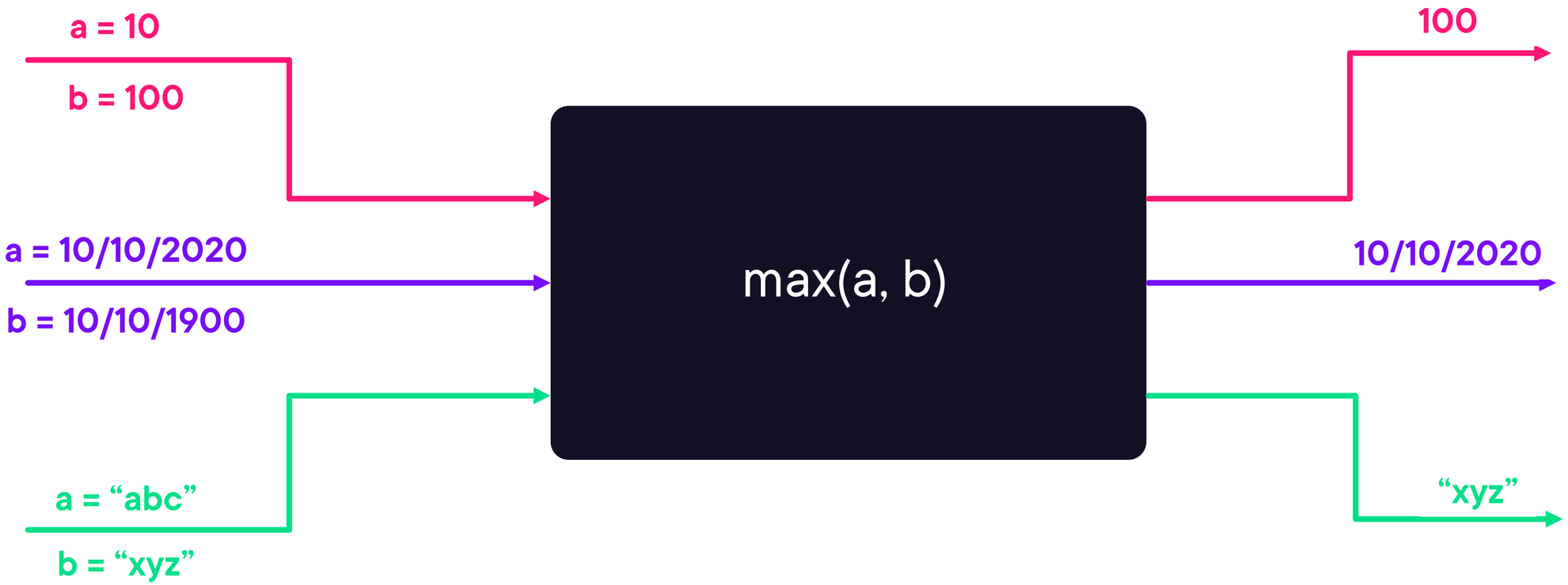
20

30

Aggregate Functions



Aggregate Functions



Types of Functions

Aggregate

Scalar

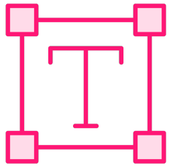
Scalar Functions

Product	UPPER(Product)
Bike	BIKE
Jacket	JACKET
Helmet	HELMET

Scalar Functions

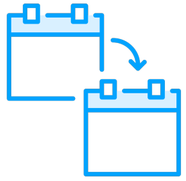
Date	YEAR(Date)	MONTH(Date)
01/01/2001	2001	1
02/02/2002	2002	2
03/03/2003	2003	3

Categories

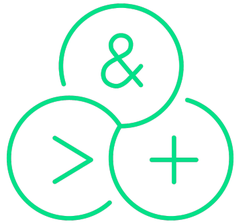


String

Mathematical



Date and time



Logical

Other

Aggregate and Scalar Functions

SELECT

ORDER BY

HAVING

Overview

Aggregate functions

Subsets

- GROUP BY
- HAVING

Sorting results

Grouping Data

Step 1: Create the Groups

Using the GROUP BY clause

Orders Table

Id	SalesTeritorry	OrderNumber	OrderDate	Amount
1	USA	16262	01012023	550
3	USA	41513	15052023	120
5	USA	59127	01012023	330
6	USA	19527	15052023	480
7	USA	68310	06062023	500

```
SELECT OrderDate, COUNT(OrderNumber) AS Orders, SUM(Amount) AS Amount
FROM Orders
WHERE SalesTeritorry = ‘USA’
GROUP BY OrderDate;
```

Orders Table

Id	SalesTeritorry	OrderNumber	OrderDate	Amount
1	USA	16262	01012023	550
3	USA	41513	15052023	120
5	USA	59127	01012023	330
6	USA	19527	15052023	480
7	USA	68310	06062023	500

```
SELECT OrderDate, COUNT(OrderNumber) AS Orders, SUM(Amount) AS Amount
FROM Orders
WHERE SalesTeritorry = 'USA'
GROUP BY OrderDate;
```

Id	SalesTeritorry	OrderNumber	OrderDate	Amount
1	USA	16262	01012023	550
5	USA	59127	01012023	330
3	USA	41513	15052023	120
6	USA	19527	15052023	480
7	USA	68310	06062023	500

```
SELECT OrderDate, COUNT(OrderNumber) AS Orders, SUM(Amount) AS Amount
FROM Orders
WHERE SalesTeritorry = 'USA'
GROUP BY OrderDate;
```

Step 2: Perform Aggregations

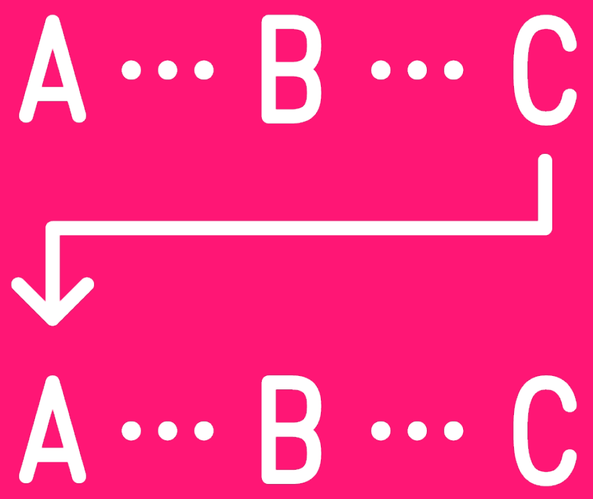
Id	SalesTeritorry	OrderNumber	OrderDate	Amount
1	USA	16262	01012023	550
5	USA	59127	01012023	330
3	USA	41513	15052023	120
6	USA	19527	15052023	480
7	USA	68310	06062023	500

```
SELECT OrderDate, COUNT(OrderNumber) AS Orders, SUM(Amount) AS Amount
FROM Orders
WHERE SalesTeritorry = 'USA'
GROUP BY OrderDate;
```

Result Table

OrderDate	Orders	Amount
01012023	2	880
15052023	2	600
06062023	1	500

```
SELECT OrderDate, COUNT(OrderNumber) AS Orders, SUM(Amount) AS Amount
FROM Orders
WHERE SalesTeritorry = ‘USA’
GROUP BY OrderDate;
```



Order is not guaranteed

All elements must be used

Demo

Use aggregate functions

- Average product price
- Least expensive product
- Most expensive product

Work with subsets of data

Filtering Groups of Data

Result Table

OrderDate	Orders	Amount
01012023	2	880000
15052023	2	600000
06062023	1	500

```
SELECT OrderDate, COUNT(OrderNumber) AS Orders, SUM(Amount) AS Amount
FROM Orders
WHERE SalesTeritorry = ‘USA’
GROUP BY OrderDate;
```

Result Table

OrderDate	Orders	Amount
01012023	2	880000
15052023	2	600000
06062023	1	500

```
SELECT OrderDate, COUNT(OrderNumber) AS Orders, SUM(Amount) AS Amount
FROM Orders
WHERE SalesTeritorry = ‘USA’ AND COUNT(OrderNumber) > 100000
GROUP BY OrderDate;
```

Result Table

OrderDate	Orders	Amount
01012023	2	880000
15052023	2	600000
06062023	1	500

```
SELECT OrderDate, COUNT(OrderNumber) AS Orders, SUM(Amount) AS Amount
FROM Orders
WHERE SalesTeritorry = ‘USA’ AND COUNT(OrderNumber) > 100000
GROUP BY OrderDate;
```

Error

Having Clause

Filters subsets of data

```
SELECT
    OrderDate,
    COUNT(OrderNumber) AS Orders,
    SUM(Amount) AS Amount
FROM Orders
WHERE SalesTeritorry = ‘USA’
GROUP BY OrderDate
HAVING COUNT(OrderNumber) > 100000;
```

Having Clause

```
SELECT
    OrderDate,
    COUNT(OrderNumber) AS Orders,
    SUM(Amount) AS Amount
FROM Orders
WHERE SalesTeritorry = ‘USA’
GROUP BY OrderDate
HAVING COUNT(OrderNumber) > 100000;
```

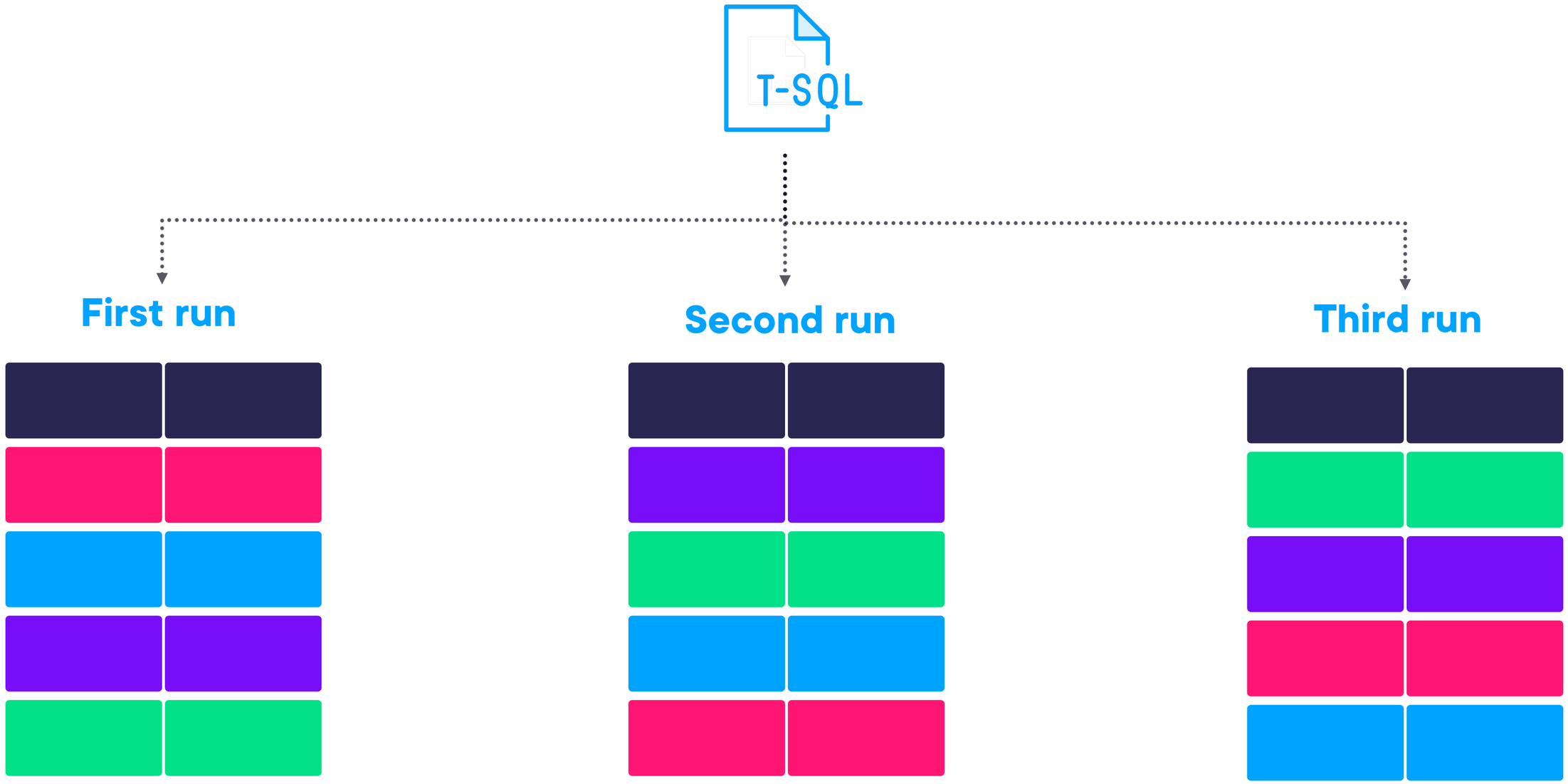
WHERE	VS	HAVING
Filters data		Filters data
Filters rows of data before grouping occurs		Filters groups of rows based on aggregation results
It's executed before the GROUP BY clause		It's executed after the GROUP BY clause
Return the rows rows for which the predicate evaluates to TRUE		Returns the groups of rows for which the predicate evaluates to TRUE
Doesn't accept aliases		Doesn't accept aliases

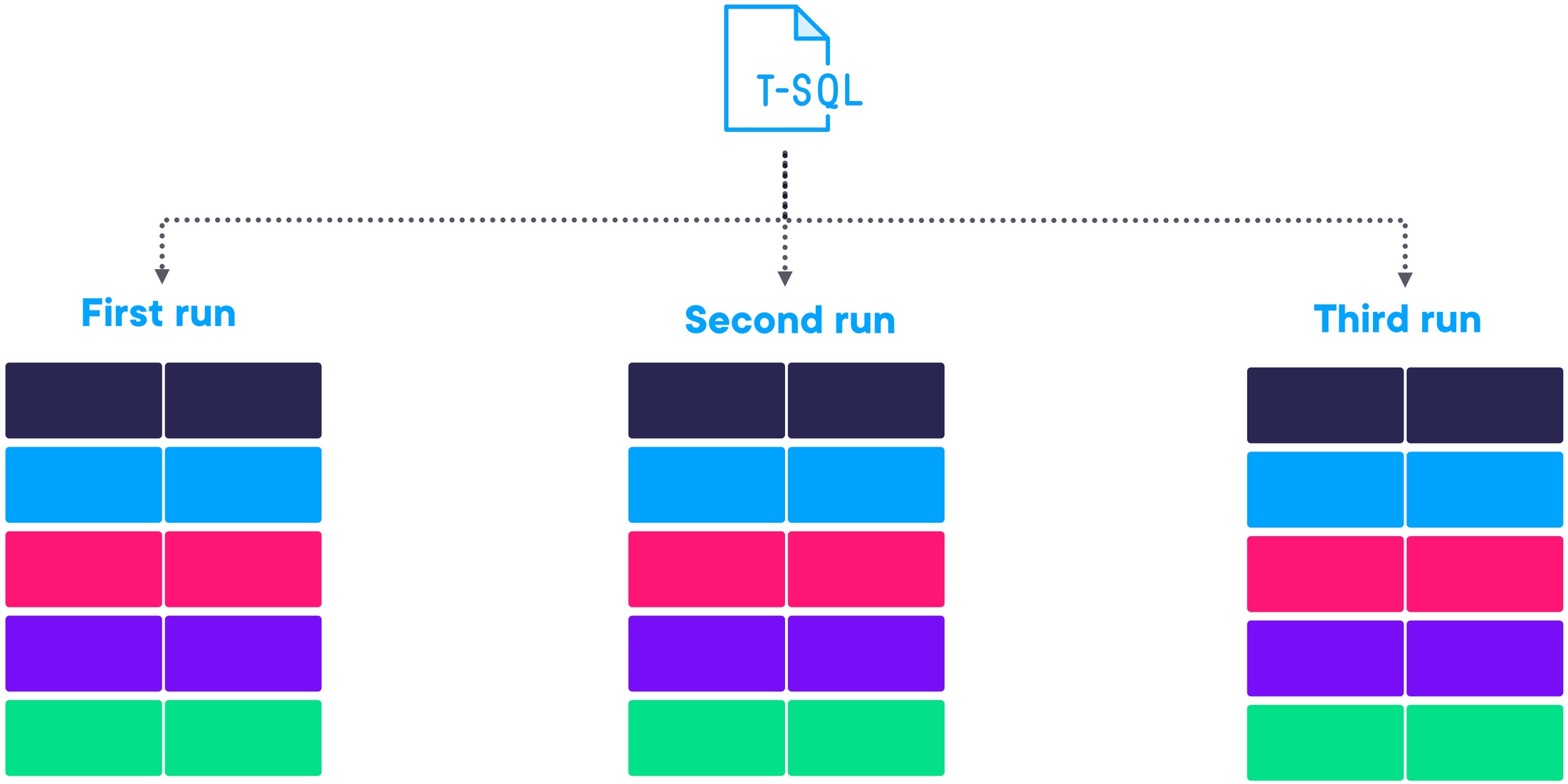
Demo

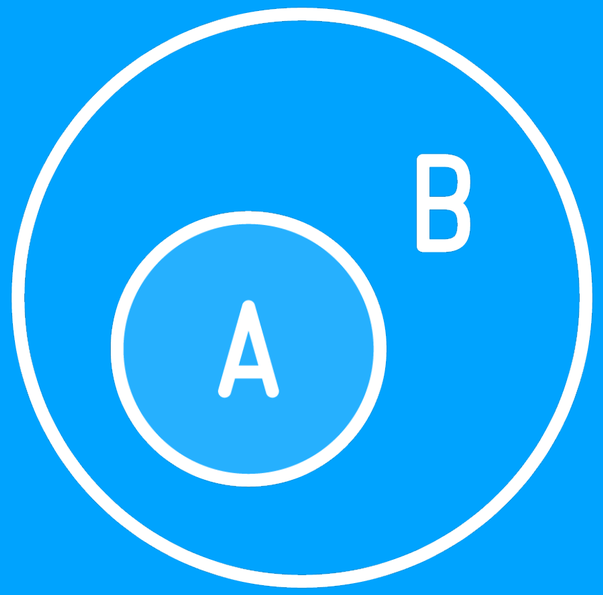
Filter the data

- Average price is greater than 100 dollars

Ordering Data


T-SQL
First run
Second run
Third run

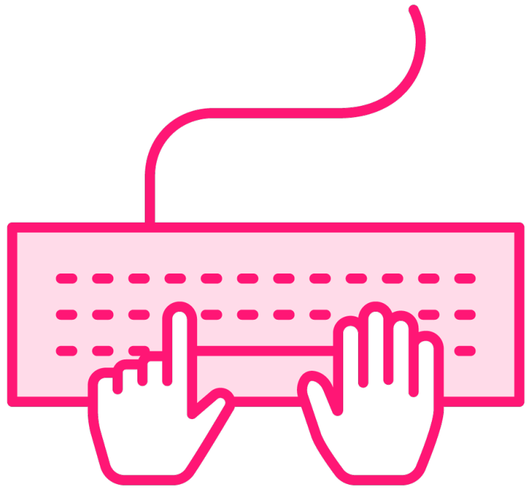

T-SQL
First run
Second run
Third run

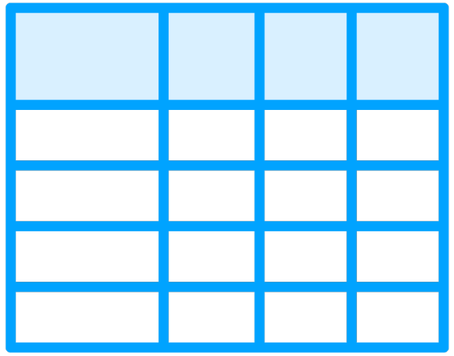


T-SQL is based on set theory.

The output of a query is a set because it lacks order.

Sets





Sets

Each element is identified by its key value.

SQL Query

```sql
SELECT OrderDate, COUNT(OrderNumber) AS Orders
FROM Orders
WHERE SalesTeritorry = 'USA'
GROUP BY OrderDate
HAVING COUNT(OrderNumber) > 1
ORDER BY OrderDate DESC, Orders;
```

FROM

WHERE

GROUP BY

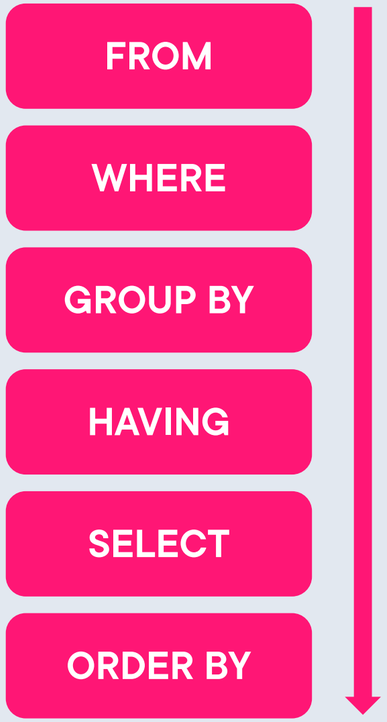
HAVING

SELECT

ORDER BY

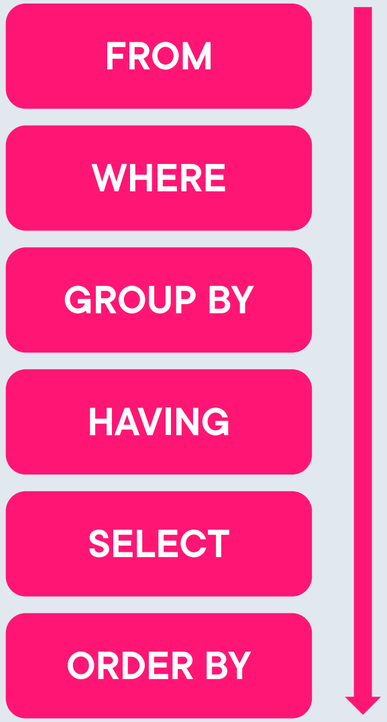
SQL Query

```
SELECT OrderDate, COUNT(OrderNumber) AS Orders
FROM Orders
WHERE SalesTeritorry = 'USA'
GROUP BY OrderDate
HAVING COUNT(OrderNumber) > 1
ORDER BY OrderDate DESC, Orders;
```



SQL Query

```sql
SELECT OrderDate, COUNT(OrderNumber) AS Orders
FROM Orders
WHERE SalesTeritorry = 'USA'
GROUP BY OrderDate
HAVING COUNT(OrderNumber) > 1
ORDER BY OrderDate DESC, Orders;
```



Sorting order

Ascending

Descending

Summary

Aggregate functions

GROUP BY clause

HAVING clause

ORDER BY clause

- Retrieved top records
- Accepts aliases

Up Next:

Joining Data