# Aggregates

- Aggregates
- User-defined Aggregates

COMP3311 20T3  $\Diamond$  Aggregates  $\Diamond$  [0/10]

### Aggregates

Aggregates reduce a collection of values into a single result.

```
Examples: count(Tuples), sum(Numbers), max(AnyOrderedType)
```

The action of an aggregate function can be viewed as:

```
State = initial state
for each item T {
    # update State to include T
    State = updateState(State, T)
}
return makeFinal(State)
```

COMP3311 20T3  $\Diamond$  Aggregates  $\Diamond$  [1/10]

# **♦ Aggregates** (cont)

Aggregates are commonly used with **GROUP BY**.

In that context, they "summarise" each group.

#### Example:

R			<pre>select a,sum(b),count(*)</pre>
a	b	С	from R group by a
+			
1	2	x	a   sum   count
1	3	У	<del>+</del> +
2	2	Z	1   5   2
2	1	a	1 5 2 2 6 3
2	2 3 2 1 3	b	

COMP3311 20T3  $\Diamond$  Aggregates  $\Diamond$  [2/10]

## User-defined Aggregates

SQL standard does not specify user-defined aggregates.

But PostgreSQL provides a mechanism for defining them.

To define a new aggregate, first need to supply:

- BaseType ... type of input values
- *StateType* ... type of intermediate states
- state mapping function: *sfunc(state,value)* → *newState*
- [optionally] an initial state value (defaults to null)
- [optionally] final function: *ffunc(state)* → *result*

<< ^ >>

### User-defined Aggregates (cont)

New aggregates defined using **CREATE AGGREGATE** statement:

- initcond (type *StateType*) is optional; defaults to **NULL**
- **finalfunc** is optional; defaults to identity function
- sortop is optional; needed for min/max-type aggregates

<< ^ >>

## **❖ User-defined Aggregates** (cont)

Example: defining the **count** aggregate (roughly)

```
create aggregate myCount(anyelement) (
    stype = int, -- the accumulator type
    initcond = 0, -- initial accumulator value
    sfunc = oneMore -- increment function
);

create function
    oneMore(sum int, x anyelement) returns int
as $$
begin return sum + 1; end;
$$ language plpgsql;
```

COMP3311 20T3 ♦ Aggregates ♦ [5/10]

Example: **sum2** sums two columns of integers

PostgreSQL has many aggregates (e.g. sum, count, ...)

But it doesn't have a product aggregate.

Implement a **prod** aggregate that

• computes the product of values in a column of numeric data

#### Usage:

```
select prod(*) from iota(5);
prod
-----
120
```

COMP3311 20T3  $\Diamond$  Aggregates  $\Diamond$  [7/10]

<< ^ >>

# **❖ User-defined Aggregates** (cont)

Example: product aggregate

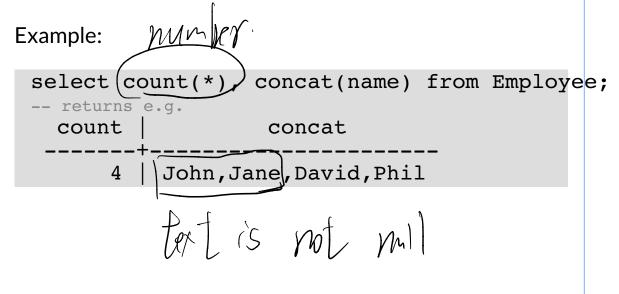
```
create function
   mult(soFar numeric, next numeric) returns numeric
as $$
begin return soFar * next; end;
$$ language plpgsql;

create aggregate prod(numeric) (
   stype = numeric,
   initcond = 1,
   sfunc = mult
);
```

COMP3311 20T3 ♦ Aggregates ♦ [8/10]

#### Define a concat aggregate that

- takes a column of string values
- returns a comma-separated string of values



COMP3311 20T3 ♦ Aggregates ♦ [9/10]

**Example:** string concatenation aggregate

COMP3311 20T3 ♦ Aggregates ♦ [10/10]

combine tert.

Produced: 15 Oct 2020

Produced: 15 Oct 2020