

## Lecture 10

- Functions
- Aggregate functions

## SQL Functions

- Most RDBMS provide a range of built-in functions to use with data manipulation statements such as SELECT, INSERT, and UPDATE
- These include numeric functions, string functions, date/time functions, flow control functions and many more – too many to cover all of them here
- All functions have in common that they return a value that we can access as a value we can SELECT.
- Some functions take arguments (parameters), and some may overloaded depending on the RDBMS (i.e., the function takes a range of different arguments)
- One basic distinction can be made between functions that operate on individual field values and "aggregate" functions that operate on groups of values
- We will discuss examples for both function types in this lecture

## A simple but useful example

- CONCAT() function for string concatenation:

```
SELECT
    CONCAT(given_name, ' ',
           surname)
FROM User;
```

```
CONCAT(given_name,'',surname)
Biao Li
Bruce Bloggs
Charlie Chucker
Claudia Cantoni
Fiona Cobber
Hine Rangit
Iosefe Johnson
Jenny Jones
Mike Tupou
Natascha Romanova
Shirley Stocker
Takahiro Kamabe
Tama Rakete
Xin Xu
```

- CONCAT() takes two or more parameters and concatenates them into a single string
- It operates on each record in the table/view, i.e., it is not an aggregate function

## Who was under 18 at the start of 2011?

- Another example, this time using the IF() function:

```
SELECT
    CONCAT(given_name, ' ',
           surname),
    date_of_birth,
    IF (date_of_birth < '1993-1-1',
        'over 18', 'under 18')
FROM User;
```

## Who was under 18 at the start of 2011?

CONCAT(given_name, ' ', surname)	date_of_birth	IF (date_of_birth < '1993-1-1', 'over 18', 'under 18')
Tama Rakete	1992-09-04 00:00:00	over 18
Hine Rangī	1992-06-21 00:00:00	over 18
Mike Tupou	1992-02-12 00:00:00	over 18
Iosefe Johnson	1991-11-12 00:00:00	over 18
Fiona Cobber	1993-04-17 00:00:00	under 18
Jenny Jones	1992-08-27 00:00:00	over 18
Takahiro Kamabe	1992-03-21 00:00:00	over 18
Charlie Chucker	1992-08-31 00:00:00	over 18
Claudia Cantoni	1991-06-19 00:00:00	over 18
Natascha Romanova	1992-02-27 00:00:00	over 18
Biao Li	1992-02-28 00:00:00	over 18
Shirley Stocker	1992-10-19 00:00:00	over 18
Xin Xu	1993-07-08 00:00:00	under 18
Bruce Bloggs	1991-11-10 00:00:00	over 18

## Aggregate functions

- The functions discussed so far all operate on each row: their return value is inserted like a field into a result set, and one value is computed for each row (record)
- But what if we want to have values computed that take in data from several records?
- Example: We want to know the number of records, or the sum or the average of a field in the records of a table
- Then we need an *aggregate* function

## Aggregate functions

- Example 1: How many users does our social network database have?

```
SELECT COUNT(*) FROM User;
```

```
COUNT(*)  
14
```

- Note that we do not have to select the count over all fields of the record, we can also select the count over a single field only, provided it isn't going to be NULL:

```
SELECT COUNT(user_id) FROM User;
```

```
COUNT(user_id)  
14
```

- Note that the end result is the same in both cases (except for the field name in the result set)

## Aggregate functions

- Example 2: How many items in total do we have in stock?

```
SELECT SUM(`Items in stock`)  
FROM stock;
```

```
SUM(`Items in stock`)  
151
```

## Aggregate functions and GROUP BY

- Often, one needs an aggregate function applied on records that have certain criteria in common.
- For example, we might want to know how many of our users live in each city:

```
SELECT COUNT(user_id), name
FROM User, City
WHERE User.city_id
      = City.city_id
GROUP BY User.city_id
```

COUNT(user_id)	name
3	Auckland
2	Wellington
2	Christchurch
1	Melbourne
1	Beijing
1	Tokyo
1	Los Angeles
1	Moscow
1	Rome
1	Singapore

## HAVING clause

- The HAVING clause is like an additional WHERE clause, but it can be used with aggregate functions and GROUP BY clauses.
- For example, if we want to know only the cities where we have more than one user, we can do this:

```
SELECT COUNT(user_id), name
FROM User, City
WHERE User.city_id
      = City.city_id
GROUP BY User.city_id
HAVING COUNT(user_id) > 1;
```

COUNT(user_id)	name
3	Auckland
2	Wellington
2	Christchurch

## Today's lab sheet

- ...is on the web
- Today: a few useful functions, aggregate and not!