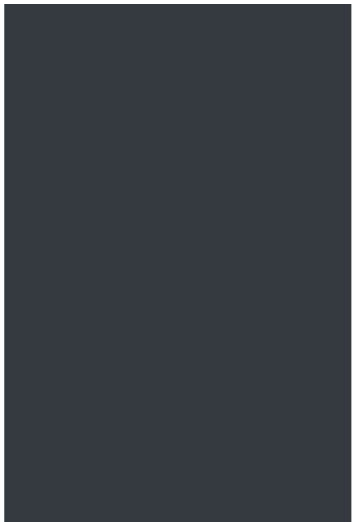
COMP3311 Week 5

Sooyoung Moon

Slides adapted from Evan Krul and Kyu-Sang Kim

Admin slides

- Assignment 1 - due 23:59 Friday 21st March 2025
- No Quiz
- Help Sessions

5	f2f **12:00—14:00**: Ainsworth Building J17 305 - Brass Lab Kenneth Li Soo-Young Moon Xinzhang Chen		f2f **14:00—16:00**: Ainsworth Building J17 G01 Kenneth Li Soo-Young Moon Xinzhang Chen Henry Li	online **10:00—13:00**: Blackboard Collaborate Soo-Young Moon Kenneth Li Neil Dadhich	online **16:00—20:00**: Blackboard Collaborate Soo-Young Moon Xinzhang Chen Abbie Worswick Kenneth Li

Pre-Tute

SQL Query Practice

SQL Views

- A “stored” query
- To make querying simpler (and more reusable)
- Return a view of the database based off some query
  - This view can then be used in other queries

SQL Views

Example: defining/naming a complex query using a view:

```
CREATE VIEW
   CourseMarksAndAverages(course,term,student,mark,avg)
AS
SELECT s.code, termName(t.id), e.student, e.mark,
       avg(mark) OVER (PARTITION BY course)
FROM   CourseEnrolments e
       JOIN Courses c on c.id = e.course
       JOIN Subjects s on s.id = c.subject
       JOIN Terms t on t.id = c.term
;
```

which would make the following query easy to solve

```
SELECT course, term, student, mark
FROM   CourseMarksAndAverages
WHERE  mark < avg;
```

plpgsql Functions

- Sometimes a view isn’t enough, sometimes we want more flexibility
- We need a **function**
  - Write a SQL function in your database
  - Query database results from a different programming language and use that language to do things (Python/psycopg2)
- plpgsql is a PostgreSQL separate procedural programming language
  - plpgsql is unlikely to be supported by almost SQL implementations except for PostgreSQL itself

plpgsql Function Syntax

```
CREATE OR REPLACE
    funcName(param1, param2, ....)
    RETURNS rettype
AS $$
DECLARE
    variable declarations
BEGIN
    code for function
END;
$$ LANGUAGE plpgsql;
```

Factorial Example

```
create or replace function
   factorial(n integer) returns integer
as $$
declare
   i integer;
   fac integer := 1;
begin
   for i in 1..n loop
      fac := fac * i;
   end loop;
   return fac;
end;
$$ language plpgsql;
```

SQL functions

SQL Functions (cont)

Differences between SQL and PLpSQL functions

- SQL function bodies are a single SQL statement
- SQL functions cannot use named parameters
  (required to use positional parameter notation: $1, $2, $3)
- SQL functions have no **`RETURN`**
  (their result is the result of the SQL statement)
- return types can be atomic, tuple, or **`setof`** tuples

```
create function add(int,int) returns int
as $$ begin return ($1 + $2); end;
$$ language plpgsql;

create function add(int,int) returns int
as $$ select $1 + $2 $$ language sql;

create function fac(n int) returns int
as $$
begin
    if (n = 0) then return 1;
    else return n * fac(n-1);
    end if;
end;
$$ language plpgsql;

create function fac(int) returns int
as $$
   select case when $1 = 0 then 1
          else $1 * fac($1-1) end
$$ language sql;
```

SQL function examples