CS2102 Lecture 5
SQL (Part 2)

Example Database

Pizzas

pizza
Diavola
Funghi
Hawaiian
Margherita
Marinara
Siciliana

Restaurants

rname	area
Corleone Corner	North
Gambino Oven	Central
Lorenzo Tavern	Central
Mamma's Place	South
Pizza King	East

Sells

rname	pizza	price
Corleone Corner	Diavola	24
Corleone Corner	Hawaiian	25
Corleone Corner	Margherita	19
Gambino Oven	Siciliana	16
Lorenzo Tavern	Funghi	23
Mamma's Place	Marinara	22
Pizza King	Diavola	17
Pizza King	Hawaiian	21

Contains

pizza	ingredient
Diavola	cheese
Diavola	chilli
Diavola	salami
Funghi	ham
Funghi	mushroom
Hawaiian	ham
Hawaiian	pineapple
Margherita	cheese
Margherita	tomato
Marinara	seafood
Siciliana	anchovies
Siciliana	capers
Siciliana	cheese

Customers

cname	area
Homer	West
Lisa	South
Maggie	East
Moe	Central
Ralph	Central
Willie	North

Likes

cname	pizza
Homer	Hawaiian
Homer	Margherita
Lisa	Funghi
Maggie	Funghi
Moe	Funghi
Moe	Sciliana
Ralph	Diavola

Simple Queries

- Basic form of SQL query consists of three clauses:

  select [**distinct**] select-list
  from from-list
  [**where** qualification]

- from-list: Specifies list of relations

- qualification: Specifies conditions on relations

- select-list: Specifies columns to be included in output table

- Output relation could contain duplicate records if **distinct** is not used in the select clause

Simple Queries (cont.)

select distinct $a_1, a_2, \cdots, a_m$

from $r_1, r_2, \cdots, r_n$

where c

$$\pi_{a_1, a_2, \ldots, a_m} \left(\sigma_c \left(r_1 \times r_2 \times \cdots \times r_n \right) \right)$$

Simple Queries (cont.)

Find the names of restaurants, the pizzas that they sell and their prices, where the price is under $20

select rname, pizza, price
from   Sells
where price < 20;

Sells

rname	pizza	price
Corleone Corner	Diavola	24
Corleone Corner	Hawaiian	25
Corleone Corner	Margherita	19
Gambino Oven	Siciliana	16
Lorenzo Tavern	Funghi	23
Mamma's Place	Marinara	22
Pizza King	Diavola	17
Pizza King	Hawaiian	21

rname	pizza	price
Corleone Corner	Margherita	19
Gambino Oven	Siciliana	16
Pizza King	Diavola	17

Simple Queries (cont.)

Find the names of restaurants, the pizzas that they sell and their prices, where the price is under $20

select rname, pizza, price
from   Sells
where price < 20;

select *
from   Sells
where price < 20;

Simple Queries (cont.)

Find all tuples from `Sells` relation such that (1) the price is under $20 and the restaurant name is not "Pizza King" or (2) the restaurant name is "Corleone Corner".

select ∗ **from** Sells
where ((price < 20) **and** (rname <> 'Pizza King'))
or   rname = 'Corleone Corner';

Sells

rname	pizza	price
Corleone Corner	Diavola	24
Corleone Corner	Hawaiian	25
Corleone Corner	Margherita	19
Gambino Oven	Siciliana	16
Lorenzo Tavern	Funghi	23
Mamma's Place	Marinara	22
Pizza King	Diavola	17
Pizza King	Hawaiian	21

rname	pizza	price
Corleone Corner	Diavola	24
Corleone Corner	Hawaiian	25
Corleone Corner	Margherita	19
Gambino Oven	Siciliana	16

Removing Duplicate Records

Q1:   **select** A, C **from** R;

Q2:   **select** **distinct** A, C **from** R;

R

A	B	C
10	1	2
10	7	2
20	3	null
20	9	null
30	3	2
30	5	9

Q1

A	C
10	2
10	2
20	null
20	null
30	2
30	9

Q2

A	C
10	2
20	null
30	2
30	9

Two tuples (a_1, c_1) and (a_2, c_2) are considered to be distinct if the following evaluates to *true*:

"(a_1 IS DISTINCT FROM a_2) or (c_1 IS DISTINCT FROM c_2)"

Expressions in SELECT Clause

select item **as** product, price $*$ qty **as** cost
from Orders;

Orders

item	price	qty
A	2.50	100
B	4.00	100
C	7.50	100

product	cost
A	250.00
B	400.00
C	750.00

Expressions in SELECT Clause (cont.)

select 'Price of ' || pizza || ' is ' || **round**(price / 1.3) || ' USD' **as** menu
from Sells
where rname = 'Corleone Corner';

Sells

rname	pizza	price
Corleone Corner	Diavola	24
Corleone Corner	Hawaiian	25
Corleone Corner	Margherita	19
Gambino Oven	Siciliana	16
Lorenzo Tavern	Funghi	23
Mamma's Place	Marinara	22
Pizza King	Diavola	17
Pizza King	Hawaiian	21

menu
Price of Diavola is 11 USD
Price of Hawaiian is 20 USD
Price of Margherita is 15 USD

|| is the string concatenation operator

`round()` = function that rounds off to nearest integer value

Set Operations

- Let Q_1 & Q_2 denote SQL queries that output union-compatible relations

- Q_1 **union** Q_2 = $Q_1 \cup Q_2$

- Q_1 **intersect** Q_2 = $Q_1 \cap Q_2$

- Q_1 **except** Q_2 = $Q_1 - Q_2$

- **union**, **intersect**, and **except** eliminate duplicate records

- **intersect** has higher precedence than **union** & **except**

    - R **union** S **intersect** T **except** V is equivalent to (R **union** (S **intersect** T)) **except** V

Set Operations (cont.)

Example 1: Find all customer/restaurant names

select cname **from** Customers

union

select rname **from** Restaurants;

Example 2: Find pizzas that contain both cheese and chilli

select pizza **from** Contains **where** ingredient = 'cheese'

intersect

select pizza **from** Contains **where** ingredient = 'chilli';

Example 3: Find pizzas that contain cheese but not chilli

select pizza **from** Contains **where** ingredient = 'cheese'

except

select pizza **from** Contains **where** ingredient = 'chilli';

Set Operations (cont.)

- **union all**, **intersect all**, and **except all** preserves duplicate records

Q1: **select** B **from** R **except** **select** B **from** S;
Q2: **select** B **from** R **except all** **select** B **from** S;

	R			S			Q1			Q2
	A	**B**		**A**	**B**		**B**			**B**
	10	1		10	1		4			1
	20	1		20	5					1
	30	1		30	2					4
	40	2		40	2					4
	50	3		50	3					
	60	4								
	70	4								

Multi-relation Queries: Example 1

Find distinct pairs of customers and restaurants that are located in the same area

select   cname, rname
from   Customers, Restaurants
where   Customers.area = Restaurants.area;

select   cname, rname
from   Customers **cross join** Restaurants
where   Customers.area = Restaurants.area;

Multi-relation Queries: Example 1 (cont.)

Find distinct pairs of customers and restaurants that are located in the same area

select	cname, rname
from	Customers **as** C, Restaurants **as** R
where	C.area = R.area;

select	cname, rname
from	Customers C, Restaurants R
where	C.area = R.area;

Multi-relation Queries: Example 1 (cont.)

Find distinct pairs of customers and restaurants that are located in the same area

select   cname, rname
from   Customers C **inner join** Restaurants R
            on C.area = R.area;

select   cname, rname
from   Customers C **join** Restaurants R
            on C.area = R.area;

Multi-relation Queries: Example 1 (cont.)

Find distinct pairs of customers and restaurants that are located in the same area

select   cname, rname
from   Customers **natural join** Restaurants;

Multi-relation Queries: Example 2

Find distinct restaurant pairs (R1,R2) where R1 $<$ R2 and they sell some common pizza

select	**distinct** S1.rname, S2.rname
from	Sells S1, Sells S2
where	S1.rname $<$ S2.rname
and	S1.pizza = S2.pizza;

select	**distinct** S1.rname, S2.rname
from	Sells S1 **join** Sells S2
	on (S1.rname $<$ S2.rname)
	and (S1.pizza = S2.pizza);

Multi-relation Queries: Example 3

Find customers and the pizzas they like; include also customers who don't like any pizza

select	C.cname, L.pizza
from	Customers C **left outer join** Likes L
	on C.cname = L.cname;

select	C.cname, L.pizza
from	Customers C **left join** Likes L
	on C.cname = L.cname;

Multi-relation Queries: Example 3 (cont.)

Find customers and the pizzas they like; include also customers who don't like any pizza

select   cname, pizza
from   Customers **natural left outer join** Likes;

select   cname, pizza
from   Customers **natural left join** Likes;

Multi-relation Queries: Example 4

Find all customer-pizza pairs (C,P) where the pizza is sold by some restaurant that is located in the same area as that of the customer. Include customers whose associated set of pizzas is empty

select   C.cname, S.pizza
from   Customers C **left join**
             (Restaurants R **join** Sells S **on** R.rname = S.rname)
             on C.area = R.area;

select   cname, pizza
from   Customers **natural left join**
             (Restaurants **natural join** Sells);

Subquery Expressions

- EXISTS subqueries

- IN subqueries

- ANY/SOME subqueries

- ALL subqueries

EXISTS Subqueries

- EXISTS (subquery)

- Returns *true* if the the output of the subquery is non-empty; otherwise, *false*

EXISTS Subqueries (cont.)

Find distinct customers who like some pizza sold by "Corleone Corner"

Likes

cname	pizza
Homer	Hawaiian
Homer	Margherita
Lisa	Funghi
Maggie	Funghi
Moe	Funghi
Moe	Sciliana
Ralph	Diavola

Sells

rname	pizza	price
Corleone Corner	Diavola	24
Corleone Corner	Hawaiian	25
Corleone Corner	Margherita	19
Gambino Oven	Siciliana	16
Lorenzo Tavern	Funghi	23
Mamma's Place	Marinara	22
Pizza King	Diavola	17
Pizza King	Hawaiian	21

cname
Homer
Ralph

EXISTS Subqueries (cont.)

Find distinct customers who like some pizza sold by "Corleone Corner"

```
select    distinct cname
from      Likes L
where     exists (
          select    1
          from      Sells S
          where     S.rname = 'Corleone Corner'
          and       S.pizza = L.pizza
          );
```

EXISTS Subqueries (cont.)

Find distinct customers who like some pizza sold by "Corleone Corner"

```
select    distinct cname
from      Likes L
where     exists (
          select    1
          from      Sells S
          where     S.rname = 'Corleone Corner'
          and       S.pizza = L.pizza
          );
```

Likes

cname	pizza
Homer	Hawaiian
Homer	Margherita
Lisa	Funghi
Maggie	Funghi
Moe	Funghi
Moe	Sciliana
Ralph	Diavola

Sells

rname	pizza	price
Corleone Corner	Diavola	24
Corleone Corner	Hawaiian	25
Corleone Corner	Margherita	19
Gambino Oven	Siciliana	16
Lorenzo Tavern	Funghi	23
Mamma's Place	Marinara	22
Pizza King	Diavola	17
Pizza King	Hawaiian	21

NOT EXISTS Subqueries

Find distinct customers who does not like any pizza sold by "Corleone Corner"

Customers

cname	area
Homer	West
Lisa	South
Maggie	East
Moe	Central
Ralph	Central
Willie	North

Likes

cname	pizza
Homer	Hawaiian
Homer	Margherita
Lisa	Funghi
Maggie	Funghi
Moe	Funghi
Moe	Sciliana
Ralph	Diavola

Sells

rname	pizza	price
Corleone Corner	Diavola	24
Corleone Corner	Hawaiian	25
Corleone Corner	Margherita	19
Gambino Oven	Siciliana	16
Lorenzo Tavern	Funghi	23
Mamma's Place	Marinara	22
Pizza King	Diavola	17
Pizza King	Hawaiian	21

cname
Lisa
Maggie
Moe
Willie

NOT EXISTS Subqueries (cont.)

Find distinct customers who does not like any pizza sold by "Corleone Corner"

select cname
from Customers C
where **not exists** (
    select 1
    from Likes L, Sells S
    where L.pizza = S.pizza
    and S.rname = 'Corleone Corner'
    and L.cname = C.cname
    );

IN Subqueries

- **expression IN (subquery)**

- Subquery must return exactly one column

- Returns *false* if the output of the subquery is empty; otherwise return the result of the boolean expression

$$((v = v_1) \text{ or } (v = v_2) \text{ or } \cdots \text{ or } (v = v_n))$$

  where

  - v denote the result of expression
  - $\{v_1, v_2, \cdots, v_n\}$ denote the output of the subquery

IN Subqueries (cont.)

Find distinct customers who like some pizza sold by "Corleone Corner"

```
select    distinct cname
from      Likes
where     pizza in (
          select     pizza
          from       Sells
          where      rname = 'Corleone Corner'
          );
```

Another Form of IN Predicate

- expression IN (value1, value2, $\cdots$, valuen)
  Example: Find pizzas that contain ham or seafood

  select   **distinct** pizza **from** Contains
  where   ingredient **in** ('ham', 'seafood');

  select   **distinct** pizza **from** Contains
  where   ingredient = 'ham' **or** ingredient = 'seafood';

  select   pizza **from** Contains **where** ingredient = 'ham'
  union
  select   pizza **from** Contains **where** ingredient = 'seafood';

ANY/SOME Subqueries

- expression operator ANY (subquery)

- Subquery must return exactly one column

- Returns *false* if the output of the subquery is empty; otherwise return the result of the boolean expression

$$((v \; op \; v_1) \text{ or } (v \; op \; v_2) \text{ or } \cdots \text{ or } (v \; op \; v_n))$$

where

  - *v* denote the result of expression
  - $\{v_1, v_2, \cdots, v_n\}$ denote the output of the subquery
  - *op* denote comparison operator (i.e., =, <>, <, >, <=, >=)

ANY/SOME Subqueries (cont.)

Find distinct restaurants that sell some pizza P1 that is more expensive than some pizza P2 sold by "Corleone Corner". P1 and P2 are not necessarily the same pizza. Exclude "Corleone Corner" from the query result.

Sells

rname	pizza	price
Corleone Corner	Diavola	24
Corleone Corner	Hawaiian	25
Corleone Corner	Margherita	19
Gambino Oven	Siciliana	16
Lorenzo Tavern	Funghi	23
Mamma's Place	Marinara	22
Pizza King	Diavola	17
Pizza King	Hawaiian	21

rname
Lorenzo Tavern
Mamma's Place
Pizza King

ANY/SOME Subqueries (cont.)

Find distinct restaurants that sell some pizza P1 that is more expensive than some pizza P2 sold by "Corleone Corner". P1 and P2 are not necessarily the same pizza. Exclude "Corleone Corner" from the query result.

```
select   distinct rname
from     Sells
where    rname <> 'Corleone Corner'
and      price > any (
         select   price
         from     Sells
         where    rname = 'Corleone Corner'
         );
```

ALL Subqueries

- expression operator ALL (subquery)

- Subquery must return exactly one column

- Returns *true* if the output of the subquery is empty; otherwise return

$$((v \; op \; v_1) \text{ and } (v \; op \; v_2) \text{ and } \cdots \text{ and } (v \; op \; v_n))$$

where

  - v denote the result of expression
  - $\{v_1, v_2, \cdots, v_n\}$ denote the output of the subquery
  - *op* denote comparison operator (i.e., =, <>, <, >, <=, >=)

ALL Subqueries (cont.)

For each restaurant, find the name and price of its most expensive pizzas. Exclude restaurants that do not sell any pizza. Assume that all prices are non-null values.

Sells

rname	pizza	price
Corleone Corner	Diavola	25
Corleone Corner	Hawaiian	25
Corleone Corner	Margherita	19
Gambino Oven	Siciliana	16
Lorenzo Tavern	Funghi	23
Mamma's Place	Marinara	22
Pizza King	Diavola	17
Pizza King	Hawaiian	21

rname	pizza	price
Corleone Corner	Diavola	25
Corleone Corner	Hawaiian	25
Gambino Oven	Siciliana	16
Lorenzo Tavern	Funghi	23
Mamma's Place	Marinara	22
Pizza King	Hawaiian	21

ALL Subqueries (cont.)

For each restaurant, find the name and price of its most expensive pizzas. Exclude restaurants that do not sell any pizza. Assume that all prices are non-null values.

```
select   rname, pizza, price
from     Sells S1
where    price >= all (
         select   S2.price
         from     Sells S2
         where    S2.rname = S1.rname
         );
```

Subqueries & Row Constructors

- So far, IN/ANY/ALL subqueries must return exactly one column

  - expression IN (subquery)
  - expression operator ANY (subquery)
  - expression operator ALL (subquery)

- Possible to use subqueries that return more than one column

  - rowConstructor IN (subquery)
  - rowConstructor operator ANY (subquery)
  - rowConstructor operator ALL (subquery)

Row Constructors

Find information on lectures that are scheduled after Wednesday 11am.

Lectures

cname	pname	day	hour
CS101	Alice	1	10
CS123	Alice	3	10
CS200	Bob	4	8
MA300	Bob	3	14

cname	pname	day	hour
CS200	Bob	4	8
MA300	Bob	3	14

select   *
from   Lectures
where   day > 3
or   ((day = 3) **and** (hour > 11));

Row Constructors (cont.)

Find information on lectures that are scheduled after Wednesday 11am.

Lectures

cname	pname	day	hour
CS101	Alice	1	10
CS123	Alice	3	10
CS200	Bob	4	8
MA300	Bob	3	14

cname	pname	day	hour
CS200	Bob	4	8
MA300	Bob	3	14

select   *
from   Lectures
where   **row**(day,hour) > **row**(3,11);

Subqueries & Row Constructors

For each lecturer, find the time of his/her earliest lecture during the week.

Lectures

cname	pname	day	hour
CS101	Alice	1	10
CS123	Alice	3	10
CS200	Bob	4	8
MA300	Bob	3	14

pname	day	hour
Alice	1	10
Bob	3	14

```
select    pname, day, hour
from      Lectures L
where     row(day,hour) <= all (
          select day, hour
          from   Lectures L2
          where L2.pname = L.pname
          );
```

Scalar Subqueries

- A scalar subquery is a subquery that returns at most one tuple with one column

  - If the subquery's output is empty, its return value is `null`

- A scalar subquery can be used as a scalar expression

Scalar Subqueries (cont.)

For each restaurant that sells Funghi, find its name, area, and selling price.

select	S.rname, R.area, S.price
from	Sells S, Restaurants R
where	S.rname = R.rname
and	S.pizza = 'Funghi';

select	rname,
	(**select** R.area **from** Restaurants R
	where R.rname = S.rname), price
from	Sells S
where	pizza = 'Funghi';

Usage of Subqueries

- Non-scalar subquery expressions can be used in different parts of SQL queries:
  - `WHERE` clause
  - `FROM` clause
  - `HAVING` clause (to be discussed in the next lecture)

Example: Subqueries in From Clause

Find distinct pizzas that are sold by restaurants
located in the east area.

```
select    distinct pizza
from      Sells natural join
          (
              select   rname
              from     Restaurants
              where    area = 'East'
          ) as East_restaurants (rname);
```

- Subqueries in from clause must be enclosed in
  parentheses & assigned a table alias. Column aliases
  are optional

Example: Subqueries in From Clause (cont.)

Lectures

cname	pname	day	hour
CS101	Alice	1	10
CS123	Alice	3	10
CS200	Bob	4	8
MA300	Bob	3	14

cname	pname	day	hour
CS101	Alice	Mon	10
CS123	Alice	Wed	10
CS200	Bob	Thu	8
MA300	Bob	Wed	14

select cname, pname, str **as** day, hour
from Lectures **natural join** (
  values (1,'Mon'), (2,'Tue'), (3,'Wed'), (4,'Thu'),
  (5,'Fri'), (6,'Sat'), (7,'Sun')
) **as** day_str (day, str);

Database Modifications with Subqueries

```
create table Students (
    studentId       integer,
    name            varchar(100),
    birthDate       date,
    year            integer,
    primary key (studentId));
```

```
create table Enrolls (
    sid     integer
            references Students,
    cid     integer
            references Courses,
    grade  char(2),
    primary key (sid, cid));
```

```
-- Enroll all first-year students in the course 101
insert into Enrolls (sid, cid)
    select studentId, 101
    from    Students
    where year = 1;
```

ORDER BY Clause

For each restaurant that sells some pizza, find its name, area, and the pizzas it sells together with their prices. Show the output in ascending order of the area, followed by in descending order of the price.

Restaurants

rname	area
Corleone Corner	North
Gambino Oven	Central
Lorenzo Tavern	Central
Mamma's Place	South
Pizza King	East

Sells

rname	pizza	price
Corleone Corner	Diavola	24
Corleone Corner	Hawaiian	25
Corleone Corner	Margherita	19
Gambino Oven	Siciliana	16
Lorenzo Tavern	Funghi	23
Mamma's Place	Marinara	22
Pizza King	Diavola	17
Pizza King	Hawaiian	21

rname	area	pizza	price
Lorenzo Tavern	Central	Funghi	23
Gambino Oven	Central	Siciliana	16
Pizza King	East	Hawaiian	19
Pizza King	East	Diavola	17
Corleone Corner	North	Hawaiian	25
Corleone Corner	North	Diavola	24
Corleone Corner	North	Margherita	19
Mamma's Place	South	Marinara	22

ORDER BY Clause (cont.)

For each restaurant that sells some pizza, find its name, area, and the pizzas it sells together with their prices. Show the output in ascending order of the area, followed by in descending order of the price.

select   *
from   Restaurants, Sells
where   Restaurants.rname = Sells.rname
order by area **asc**, price **desc**;

select   *
from   Restaurants, Sells
where   Restaurants.rname = Sells.rname
order by area, price **desc**;

LIMIT Clause

Find the top three most expensive pizzas. Show the pizza name, the name of the restaurant that sells it, and its selling price for each output tuple; and show the output in descending order of price.

Sells

rname	pizza	price
Corleone Corner	Diavola	24
Corleone Corner	Hawaiian	25
Corleone Corner	Margherita	19
Gambino Oven	Siciliana	16
Lorenzo Tavern	Funghi	23
Mamma's Place	Marinara	22
Pizza King	Diavola	17
Pizza King	Hawaiian	21

pizza	rname	price
Hawaiian	Corleone Corner	25
Diavola	Corleone Corner	24
Funghi	Lorenzo Tavern	23

select   pizza, rname, price
from   Sells
order by   price **desc**
limit   3;

OFFSET Clause

For each pizza that is sold by some restaurant, find the pizza name, the restaurant name and its selling price; show the output in descending order of price and exclude the top three pizzas.

Sells

rname	pizza	price
Corleone Corner	Diavola	24
Corleone Corner	Hawaiian	25
Corleone Corner	Margherita	19
Gambino Oven	Siciliana	16
Lorenzo Tavern	Funghi	23
Mamma's Place	Marinara	22
Pizza King	Diavola	17
Pizza King	Hawaiian	21

pizza	rname	price
Mamma's Place	Marinara	22
Pizza King	Hawaiian	21
Corleone Corner	Margherita	19
Pizza King	Diavola	17
Gambino Oven	Siciliana	16

select	pizza, rname, price
from	Sells
order by	price **desc**
offset	3;

OFFSET Clause

Assume that Sells.price is unique. Find the 4th and 5th most expensive pizzas. Show the pizza name, the name of the restaurant that sells it, and its selling price for each output tuple; and show the output in descending order of price.

Sells

rname	pizza	price
Corleone Corner	Diavola	24
Corleone Corner	Hawaiian	25
Corleone Corner	Margherita	19
Gambino Oven	Siciliana	16
Lorenzo Tavern	Funghi	23
Mamma's Place	Marinara	22
Pizza King	Diavola	17
Pizza King	Hawaiian	21

pizza	rname	price
Mamma's Place	Marinara	22
Pizza King	Hawaiian	21

select pizza, rname, price
from Sells
order by price **desc**
offset 3
limit 2;

SQL:2008 Syntax for LIMIT & OFFSET

select	pizza, rname, price
from	Sells
order by	price **desc**
offset	3
limit	2;

select	pizza, rname, price
from	Sells
order by	price **desc**
offset	3 **rows**
fetch next	2 **rows only**;

Summary

- Basic SQL queries

select	**distinct** select-list
from	from-list
where	where-condition
order by	orderby-list
offset	offset-specification
limit	limit-specification

- Non-scalar subqueries can be used in `FROM`, `WHERE`, and `HAVING` clauses

- SQL Reference: `https://www.postgresql.org/docs/current/index.html`