# SQL

types
- char(n): Fixed
- varchar(n): Variable
- int
- smallint
- numeric(p,d): Fixed point (3,1) allows 44.5 not 444.8
- real, double precision: Floating + Double-precision
- float(n)

CREATE TABLE

create table name ( ID char(5), name varchar(20), salary numeric(8,2));

Not Null
- primary key (ID), foreign key (name) references department);

primary key declaration is NOT NULL.

UPDATING TABLE
- Insert into _table_ values (_,_,_)
- delete from student
- drop table r
- alter table r add A D
  alter table r drop D

Query Structure

Select $A_1, A_2 \ldots A_n$ from $r_1, r_2, \ldots, r_m$ where P

($A_1, A_2 \ldots A_n$: Attribute (column); $r_1, r_2, \ldots, r_m$: relation (tables); P: predicate)

Returns table (relation)

Select * from table

(* = ALL)

Arth. Expressions

Select ID, Name, salary/12 from instructor

Return column salary w salary/12

Can ReName

Select ID, Name, salary/12 as monthly_Salary from table

Where

query can be combined (and, or, not)

Select name
from table
where query

```
select name
from instructor
where dept_name = 'Comp. Sci.'
```

String Operations

'%' matches substring

'_' character matches any character

'\' escape character

'Intro%' matches any string beginning with "Intro".

'%Comp%' matches any string containing "Comp" as a substring.

'___' matches any string of exactly three characters.

'___%' matches any string of at least three characters.

Ordering

select distinct column
from table
order by column desc/asc

(desc = descending, asc = acending)

multiple values: dep_name, name, etc

# Where

Select name
from table
where salary between 90000 and 100000

(between ... and ... : query)

# Set Operations

Find courses that ran in Fall 2009 or in Spring 2010

(**select** *course_id* **from** *section* **where** *sem* = 'Fall' **and** *year* = 2009)
**union**
(**select** *course_id* **from** *section* **where** *sem* = 'Spring' **and** *year* = 2010)

Find courses that ran in Fall 2009 and in Spring 2010

(**select** *course_id* **from** *section* **where** *sem* = 'Fall' **and** *year* = 2009)
**intersect**
(**select** *course_id* **from** *section* **where** *sem* = 'Spring' **and** *year* = 2010)

Find courses that ran in Fall 2009 but not in Spring 2010

(**select** *course_id* **from** *section* **where** *sem* = 'Fall' **and** *year* = 2009)
**except**
(**select** *course_id* **from** *section* **where** *sem* = 'Spring' **and** *year* = 2010)

Find the salaries of all instructors that are less than the largest salary.

- **select distinct** *T.salary*
  **from** *instructor* **as** *T*, *instructor* **as** *S*
  **where** *T.salary* < *S.salary*

Find all the salaries of all instructors

- **select distinct** *salary*
  **from** *instructor*

Find the largest salary of all instructors.

- (**select** "second query")
  **except**
  (**select** "first query")

# All

- add all after set operation to retain duplicates

# Select Null Values

Select name
from table
where Salary is null

Any comparison with *null* returns *unknown*

- Example: 5 < *null* or *null* <> *null* or *null* = *null*

Three-valued logic using the value *unknown*:

- OR: (*unknown* **or** *true*) = *true*,
  (*unknown* **or** *false*) = *unknown*
  (*unknown* **or** *unknown*) = *unknown*
- AND: (*true* **and** *unknown*) = *unknown*,
  (*false* **and** *unknown*) = *false*,
  (*unknown* **and** *unknown*) = *unknown*
- NOT: (**not** *unknown*) = *unknown*
- "*P* **is unknown**" evaluates to true if predicate *P* evaluates to *unknown*

where + unknown = false

# Functions

avg
min
max
sum
count

Select FN (column)
from table
where query

# Subqueries From Clause

SQL allows a subquery expression to be used in the **from** clause

Find the average instructors' salaries of those departments where the average salary is greater than $42,000.

```
select dept_name, avg_salary
from (select dept_name, avg (salary) as avg_salary
      from instructor
      group by dept_name)
where avg_salary > 42000;
```

Note that we do not need to use the **having** clause

Another way to write above query

```
select dept_name, avg_salary
from (select dept_name, avg (salary)
      from instructor
      group by dept_name) as dept_avg (dept_name, avg_salary)
where avg_salary > 42000;
```

# with

The **with** clause provides a way of defining a temporary relation whose definition is available only to the query in which the **with** clause occurs.

Find all departments with the maximum budget

```
with max_budget (value) as
    (select max(budget)
     from department)
select department.name
from department, max_budget
where department.budget = max_budget.value;
```

```
with dept_total (dept_name, value) as
    (select dept_name, sum(salary)
     from instructor
     group by dept_name),
  dept_total_avg(value) as
    (select avg(value)
     from dept_total)
select dept_name
from dept_total, dept_total_avg
where dept_total.value > dept_total_avg.value;
```

Jack - 3556 - 2024

# Having Clause

- applied after formating groups

Find the names and average salaries of all departments whose average salary is greater than 42000

```
select dept_name, avg (salary)
from instructor
group by dept_name
having avg (salary) > 42000;
```

# Group by

Select ____
from table
group by column

Find the average salary of instructors in each department

```
select dept_name, avg (salary) as avg_salary
from instructor
group by dept_name;
```

| ID | name | dept_name | salary |
|---|---|---|---|
| 76766 | Crick | Biology | 72000 |
| 45565 | Katz | Comp. Sci. | 75000 |
| 10101 | Srinivasan | Comp. Sci. | 65000 |
| 83821 | Brandt | Comp. Sci. | 92000 |
| 98345 | Kim | Elec. Eng. | 80000 |
| 12121 | Wu | Finance | 90000 |
| 76543 | Singh | Finance | 80000 |
| 32343 | El Said | History | 60000 |
| 58583 | Califieri | History | 62000 |
| 15151 | Mozart | Music | 40000 |
| 33456 | Gold | Physics | 87000 |
| 22222 | Einstein | Physics | 95000 |

| dept_name | avg_salary |
|---|---|
| Biology | 72000 |
| Comp. Sci. | 77333 |
| Elec. Eng. | 80000 |
| Finance | 85000 |
| History | 61000 |
| Music | 40000 |
| Physics | 91000 |

# Subqueries in Select Clause

Scalar subquery is one which is used where a single value is expected

List all departments along with the number of instructors in each department

```
select dept_name,
       (select count(*)
        from instructor
        where department.dept_name = instructor.dept_name)
       as num_instructors
from department;
```

Runtime error if subquery returns more than one result tuple

# Database Modification

- delete from table
  where query in (select column
  from table
  where query)
- insert into table
  values (_,_,_,_);

  insert into table
  Select column, column..., Column, 0 *
  from table

  (NO, it just adds column w All Zeros)

  (* = value at least column that expected)

Add all table with column set to 0 into table

The **select from where** statement is evaluated fully before any of its results are inserted into the relation.

Otherwise queries like

```
insert into table1 select * from table1
```

would cause problem

← can Do this

## Updates

Increase salaries of instructors whose salary is over $100,000 by 3%, and all others by a 5%

- Write two **update** statements:

```
update instructor
  set salary = salary * 1.03
  where salary > 100000;
update instructor
  set salary = salary * 1.05
  where salary <= 100000;
```

- The order is important
- Can be done better using the **case** statement (next slide)

# Cases

Same query as before but with case statement

```
update instructor
  set salary = case
               when salary <= 100000 then salary * 1.05
               else salary * 1.03
               end
```