String Operations

- for comparisons on character strings. The operator “like” uses patterns that are described using two special characters:
  - percent (%). matches any substring.
  - underscore (_). matches any character.
- Find names of all instructors whose name includes the substring “dar”.

```
select name
from instructor
where name like '%dar%'
```

- For “100 %” : like ‘100 \%' using escape '\‘
- String concatenation using ||
- Patterns are case sensitive

Ordering the Query Result

- List in alphabetic order the names of all instructors :

```
select distinct name
  from   instructor
  order by name
```

- Use desc for descending order or asc for ascending order; ascending is the default.
- Can sort on multiple attributes
  - Example: order by *dept_name desc, name asc*

more Where Clause Predicates

- between clause

  where *salary* between 90000 and 100000

  <both inclusive>

- Tuple comparison

```
select name, course_id
from instructor, teaches
where (instructor.ID, dept_name) =
                      (teaches.ID, 'Biology');
```

Set Operations

- On union-compatible relations
- 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)
```

□ **intersect** and **except** (for difference) also provided

These automatically eliminate duplicates
To retain all duplicates use union all, intersect all and except all.

Null Values

- Some attributes may have null values
- *null* signifies unknown or not applicable.
- The result of any arithmetic expression involving *null* is *null*
  - Example: 5 + *null* returns null
- is null can be used to check for null values.

```
select name
from instructor
where salary is null
```

- Result of where clause predicate is treated as *false* if it evaluates to *unknown*

Null Values and Three Valued Logic

- Any comparison with *null* returns *unknown*
  - Example*: 5 < null   or   null <> null   or   null = null*
- Three-valued logic : true, false, *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*

The condition in where should be true for data to be selected in the result

Aggregate Functions

- These functions operate on the values of a column of a relation, and return a value

avg	average value
min	minimum value
max	maximum value
sum	sum of values
count	number of values

- Aggregation may be done on
  - Whole table
  - Result of a query
  - Breaking a table/result into groups

Aggregate Functions (Cont.)

- Find average salary of instructors in the Computer Science

```
select avg (salary)
 from instructor
 where dept_name= 'Comp. Sci.';
```

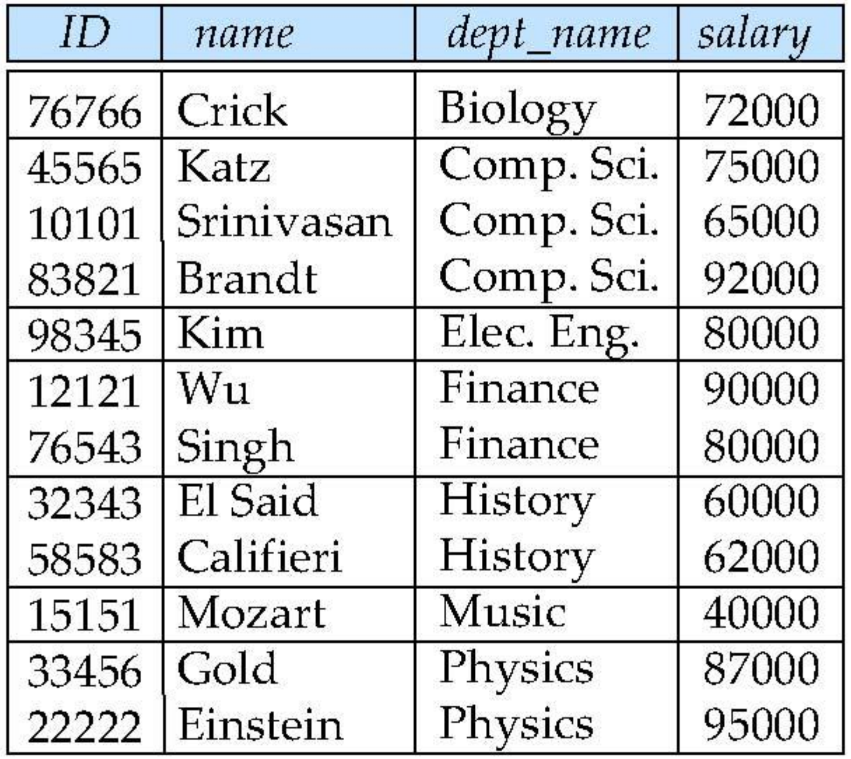
- Find the total number of instructors who teach a course in the Spring 2010 semester

```
select count (distinct ID)
 from teaches
 where semester = 'Spring' and year = 2010
```

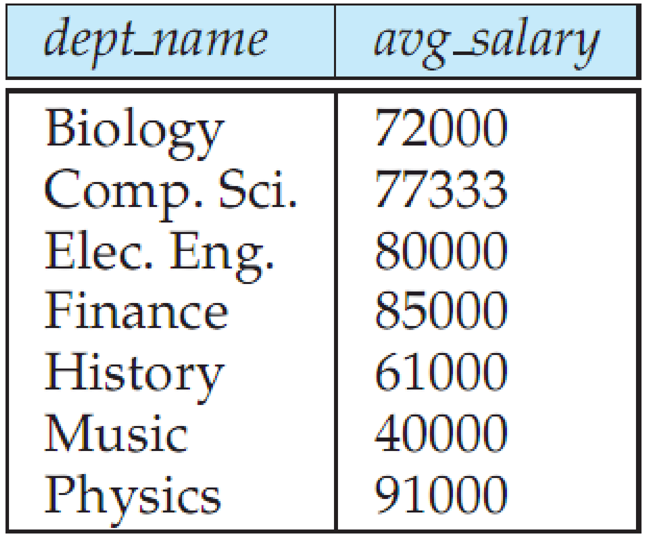
Aggregate Functions – Group By

- Find average salary of instructors in each department

```
select dept_name, 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

Aggregation (Cont.)

- With grouping, the SELECT may contain grouping attributes and aggregate functions
- Cannot contain other attributes

```
/* erroneous query */
select dept_name, ID, avg (salary)
from instructor
group by dept_name;
```

Aggregate Functions – Having Clause

- Find 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;
```

predicates in the having clause are applied after the formation of groups whereas predicates in the where clause are applied before forming groups

Null Values and Aggregates

- Total all salaries

```
select sum (salary )
from instructor
```

  - Above statement ignores null amounts
  - Result is *null* if there is no non-null amount
- All aggregate operations except count(*) ignore tuples with null values on the aggregated attributes
- What if collection has only null values?
  - count returns 0
  - all other aggregates return null