# SQL- Structured Query Language

## Database management system

Elinor Brondwine

Based on lecture slides and book material by R.Ramakrishnan and J.Gehrke, Database Management System 3ed

Universität Regensburg

# Previous lectures

- Entity relationship diagrams
  - Requirements
  - Modeling
  - Touching upon design
- The relational model
  - Tables and schemas
  - From ERD to a relational DBMS
- Basic SQL
  - Creating/deleting/modifying tables
  - Inserting data
  - Integrity constraints
- Relational query language

# Previous lectures

- Entity relationship diagrams
  - Requirements
  - Modeling
  - Touching upon design
- The relational model
  - Tables and schemas
  - From ERD to a relational DBMS
- Basic SQL
  - Creating/deleting/modifying  tables
  - Inserting data
  - Integrity constraints
- Relational query language

# Next: SQL Queries

- Query components:
    - SELECT
    - FROM
    - WHERE
    - GROUP BY
    - HAVING
    - ORDER BY

# Example from previous lecture

- Find the names of the voyagers who are listed to an excursion to Venice

E

| tid | vid | excursion |
|---|---|---|
| 95 | 345 | Rome |
| 25 | 856 | Munich |
| 95 | 345 | Venice |
| 95 | 123 | Venice |

V

| vid | vname | experience |
|---|---|---|
| 123 | Schwartz | 4 |
| 345 | Weiss | 4 |
| 856 | Weber | 2 |

# Answer:

$$\pi_{vname}\left(\sigma_{(E.vid=V.vid)\hat{}(excursion='Venice')}(E \times V)\right)$$

E

| tid | vid | excursion |
|---|---|---|
| 95 | 345 | Rome |
| 25 | 856 | Munich |
| 95 | 345 | Venice |
| 95 | 123 | Venice |

V

| vid | vname | experience |
|---|---|---|
| 123 | Schwartz | 4 |
| 345 | Weiss | 4 |
| 856 | Weber | 2 |

# Answer:

$$\pi_{vname}\left(\sigma_{(E.vid=V.vid){}^\wedge(excursion='Venice')}(E \times V)\right)$$

| V.experience | V.vname | V.vid | E.excursion | E.vid | E.tid |
|---|---|---|---|---|---|
| 4 | Schwartz | 123 | Rome | 345 | 95 |
| 4 | Schwartz | 123 | Munich | 856 | 25 |
| 4 | Schwartz | 123 | Venice | 345 | 95 |
| 4 | Schwartz | 123 | Venice | 123 | 95 |
| 4 | Weiss | 345 | Rome | 345 | 95 |
| 4 | Weiss | 345 | Munich | 856 | 25 |
| 4 | Weiss | 345 | Venice | 345 | 95 |
| 4 | Weiss | 345 | Venice | 123 | 95 |
| 2 | Weber | 856 | Rome | 345 | 95 |
| 2 | Weber | 856 | Munich | 856 | 25 |
| 2 | Weber | 856 | Venice | 345 | 95 |

# Answer:

$$\pi_{vname}\left(\sigma_{(E.vid=V.vid)\wedge(excursion='Venice')}(E\times V)\right)$$

| V.experience | V.vname | V.vid | E.excursion | E.vid | E.tid |
|---|---|---|---|---|---|
| 4 | Schwartz | 123 | Rome | 345 | 95 |
| 4 | Schwartz | 123 | Munich | 856 | 25 |
| 4 | Schwartz | 123 | Venice | 345 | 95 |
| 4 | Schwartz | 123 | Venice | 123 | 95 |
| 4 | Weiss | 345 | Rome | 345 | 95 |
| 4 | Weiss | 345 | Munich | 856 | 25 |
| 4 | Weiss | 345 | Venice | 345 | 95 |
| 4 | Weiss | 345 | Venice | 123 | 95 |
| 2 | Weber | 856 | Rome | 345 | 95 |
| 2 | Weber | 856 | Munich | 856 | 25 |
| 2 | Weber | 856 | Venice | 345 | 95 |

# Answer:

$$\pi_{vname}\left(\sigma_{(E.vid=V.vid)\wedge(excursion='Venice')}(E \times V)\right)$$

| V.experience | V.vname | V.vid | E.excursion | E.vid | E.tid |
|---|---|---|---|---|---|
| 4 | Schwartz | 123 | Rome | 345 | 95 |
| 4 | Schwartz | 123 | Munich | 856 | 25 |
| 4 | Schwartz | 123 | Venice | 345 | 95 |
| 4 | Schwartz | 123 | Venice | 123 | 95 |
| 4 | Weiss | 345 | Rome | 345 | 95 |
| 4 | Weiss | 345 | Munich | 856 | 25 |
| 4 | Weiss | 345 | Venice | 345 | 95 |
| 4 | Weiss | 345 | Venice | 123 | 95 |
| 2 | Weber | 856 | Rome | 345 | 95 |
| 2 | Weber | 856 | Munich | 856 | 25 |
| 2 | Weber | 856 | Venice | 345 | 95 |

# Answer:

$$\pi_{vname}\left(\sigma_{(E.vid=V.vid)^\wedge(excursion='Venice')}(E \times V)\right)$$

| V.experience | V.vname | V.vid | E.excursion | E.vid | E.tid |
|---|---|---|---|---|---|
| 4 | Schwartz | 123 | Rome | 345 | 95 |
| 4 | Schwartz | 123 | Munich | 856 | 25 |
| 4 | Schwartz | 123 | Venice | 345 | 95 |
| 4 | Schwartz | 123 | Venice | 123 | 95 |
| 4 | Weiss | 345 | Rome | 345 | 95 |
| 4 | Weiss | 345 | Munich | 856 | 25 |
| 4 | Weiss | 345 | Venice | 345 | 95 |
| 4 | Weiss | 345 | Venice | 123 | 95 |
| 2 | Weber | 856 | Rome | 345 | 95 |
| 2 | Weber | 856 | Munich | 856 | 25 |
| 2 | Weber | 856 | Venice | 345 | 95 |

# Answer:

$$\pi_{vname}\left(\sigma_{(E.vid=V.vid)\wedge(excursion='Venice')}(E \times V)\right)$$

| E.tid | E.vid | E.excursion | V.vid | V.vname | V.experience |
|---|---|---|---|---|---|
| 95 | 123 | Venice | 123 | Schwartz | 4 |
| 95 | 345 | Venice | 345 | Weiss | 4 |

# Answer:

$$\pi_{vname}\left(\sigma_{(E.vid=V.vid)^\wedge(excursion='Venice')}(E \times V)\right)$$

| E.tid | E.vid | E.excursion | V.vid | V.vname | V.experience |
|---|---|---|---|---|---|
| 95 | 123 | Venice | 123 | Schwartz | 4 |
| 95 | 345 | Venice | 345 | Weiss | 4 |

# Answer:

$$\pi_{vname}\left(\sigma_{(E.vid=V.vid)^\wedge(excursion='Venice')}(E \times V)\right)$$

| V.vname |
|---|
| Schwartz |
| Weiss |

# Answer:

$$\pi_{vname}\left(\sigma_{(E.vid=V.vid)^\wedge(excursion='Venice')}(E\times V)\right)$$

| V.vname |
|---|
| Schwartz |
| Weiss |

# Answer:

$$\pi_{vname}\left(\sigma_{(E.vid=V.vid)\hat{}(excursion='Venice')}(E\times V)\right)$$

E

| tid | vid | excursion |
|---|---|---|
| 95 | 345 | Rome |
| 25 | 856 | Munich |
| 95 | 345 | Venice |
| 95 | 123 | Venice |

V

| vid | vname | experience |
|---|---|---|
| 123 | Schwartz | 4 |
| 345 | Weiss | 4 |
| 856 | Weber | 2 |

# Equivalent SQL query

```
SELECT DISTINCT V.vname
  FROM E,V
 WHERE E.vid=V.vid and E.excursion=‘Venice’;
```

# Basic SQL Queries

$$\pi_{A_1,\ldots,A_n}(R_1 \times \cdots \times R_m)$$

# Basic SQL Queries

$$\pi_{A_1,\ldots,A_n}(R_1 \times \cdots \times R_m)$$

```
SELECT DISTINCT A1,…,An
   FROM R1,…,Rm;
```

# Basic SQL Queries

$$\pi_{A_1,\ldots,A_n}\left(\sigma_C(R_1 \times \cdots \times R_m)\right)$$

# Basic SQL Queries

$$\pi_{A_1,\ldots,A_n}\left(\sigma_C(R_1 \times \cdots \times R_m)\right)$$

```
SELECT DISTINCT A1,…,An
  FROM R1,…,Rm
 WHERE C;
```

# Basic SQL Queries

$R_1 \times \cdots \times R_m$

# Basic SQL Queries

$$R_1 \times \cdots \times R_m$$

```
SELECT DISTINCT *
   FROM R1,…,Rm;
```

# Basic SQL Queries

$\sigma_C(R_1 \times \cdots \times R_m)$

# Basic SQL Queries

$\sigma_C(R_1 \times \cdots \times R_m)$

```
SELECT DISTINCT *
  FROM R1,…,Rm
 WHERE C;
```

# Basic SQL Query

```
SELECT [DISTINCT] target-list
  FROM relation-list
 [WHERE condition];
```

- `relation-list`: list of relation names
- `target-list`: list of attributes onto which the output relation is projected
- Condition: optional Boolean condition

# Basic SQL Query

```
SELECT [DISTINCT] target-list
  FROM relation-list
 [WHERE condition];
```

- `relation-list`: list of relation names
- `target-list`: list of attributes onto which the output relation is projected
- `Condition`: optional Boolean condition

# Basic SQL Query

```
SELECT [DISTINCT] target-list
  FROM relation-list
 [WHERE condition];
```

- `relation-list`: list of relation names
- `target-list`: list of attributes onto which the output relation is projected
- `Condition`: optional Boolean condition

# Basic SQL Query

```
SELECT [DISTINCT] target-list
  FROM relation-list
 [WHERE condition];
```

- `relation-list`: list of relation names
- `target-list`: list of attributes onto which the output relation is projected
- `Condition`: optional Boolean condition

# Basic SQL Query

```
SELECT [DISTINCT] target-list
  FROM relation-list
 [WHERE condition];
```

**Query evaluation:**

1. Compute the cross-product of *relation-list*
2. Discard resulting tuples if they fail *condition*
3. Delete attributes that are not in *target-list*
4. If DISTINCT is specified, eliminate duplicate rows.

# Basic SQL Query

```
SELECT [DISTINCT] target-list
  FROM relation-list
 [WHERE condition];
```

**Query evaluation:**

1. Compute the cross-product of *relation-list*
2. Discard resulting tuples if they fail *condition*
3. Delete attributes that are not in *target-list*
4. If DISTINCT is specified, eliminate duplicate rows.

# Basic SQL Query

```
SELECT [DISTINCT] target-list
  FROM relation-list
 [WHERE condition];
```

**Query evaluation:**

1. Compute the cross-product of *relation-list*
2. Discard resulting tuples if they fail *condition*
3. Delete attributes that are not in *target-list*
4. If DISTINCT is specified, eliminate duplicate rows.

# Basic SQL Query

```
SELECT [DISTINCT] target-list
  FROM relation-list
 [WHERE condition];
```

**Query evaluation:**

1. Compute the cross-product of *relation-list*
2. Discard resulting tuples if they fail *condition*
3. Delete attributes that are not in *target-list*
4. If DISTINCT is specified, eliminate duplicate rows.

# Basic SQL Query

```
SELECT [DISTINCT] target-list
  FROM relation-list
 [WHERE condition];
```

**Query evaluation:**

1. Compute the cross-product of *relation-list*
2. Discard resulting tuples if they fail *condition*
3. Delete attributes that are not in *target-list*
4. If DISTINCT is specified, eliminate duplicate rows.

# Basic SQL Query

```
SELECT [DISTINCT] target-list
  FROM relation-list
 [WHERE condition];
```

**This strategy is probably the least efficient way to compute a query! An optimizer will find more efficient strategies to compute the same answers..**

# Example Relations

| Sailors | | | |
|---|---|---|---|
| sid | sname | rating | age |
| 22 | Dustin | 7 | 45.0 |
| 31 | Lubber | 8 | 55.5 |
| 58 | Rusty | 10 | 35.0 |

| Boats | | |
|---|---|---|
| bid | bname | color |
| 101 | Interlake | blue |
| 103 | Clipper | green |

| Reserves | | |
|---|---|---|
| sid | bid | day |
| 22 | 101 | 10/10/96 |
| 58 | 103 | 11/12/96 |

# Find all sailors with a rating above 7

# Find all sailors with a rating above 7

$\sigma_{rating>7}(Sailors)$

# Find all sailors with a rating above 7

$$\sigma_{rating>7}(Sailors)$$

```
SELECT DISTINCT *
  FROM Sailors
 WHERE rating>7;
```

# Names of sailors with a rating above 7

# Example Relations

| Sailors | | | |
|---|---|---|---|
| sid | sname | rating | age |
| 22 | Dustin | 7 | 45.0 |
| 31 | Lubber | 8 | 55.5 |
| 58 | Rusty | 10 | 35.0 |

| Boats | | |
|---|---|---|
| bid | bname | color |
| 101 | Interlake | blue |
| 103 | Clipper | green |

| Reserves | | |
|---|---|---|
| sid | bid | day |
| 22 | 101 | 10/10/96 |
| 58 | 103 | 11/12/96 |

# Names of sailors with a rating above 7

```
SELECT DISTINCT sname
  FROM Sailors
 WHERE rating>7;
```

$$\pi_{sname}(\sigma_{rating>7}(Sailors))$$

# Find the names of sailors who have reserved boat number 103

```
SELECT [DISTINCT] target-list
  FROM relation-list
 [WHERE condition];
```

| Sailors | | | |
|---|---|---|---|
| sid | sname | rating | age |
| 22 | Dustin | 7 | 45.0 |
| 31 | Lubber | 8 | 55.5 |
| 58 | Rusty | 10 | 35.0 |

| Reserves | | |
|---|---|---|
| sid | bid | day |
| 22 | 101 | 10/10/96 |
| 58 | 103 | 11/12/96 |

# Find the names of sailors who have reserved boat number 103

```
SELECT [DISTINCT] target-list
   FROM
 [WHERE condition];
```

| Sailors | | | |
|---|---|---|---|
| sid | sname | rating | age |
| 22 | Dustin | 7 | 45.0 |
| 31 | Lubber | 8 | 55.5 |
| 58 | Rusty | 10 | 35.0 |

| Reserves | | |
|---|---|---|
| sid | bid | day |
| 22 | 101 | 10/10/96 |
| 58 | 103 | 11/12/96 |

# Find the names of sailors who have reserved boat number 103

```
SELECT [DISTINCT] target-list
   FROM  Sailors, Reserves
  [WHERE condition];
```

| Sailors | | | |
|---|---|---|---|
| sid | sname | rating | age |
| 22 | Dustin | 7 | 45.0 |
| 31 | Lubber | 8 | 55.5 |
| 58 | Rusty | 10 | 35.0 |

| Reserves | | |
|---|---|---|
| sid | bid | day |
| 22 | 101 | 10/10/96 |
| 58 | 103 | 11/12/96 |

# Find the names of sailors who have reserved boat number 103

```
SELECT [DISTINCT] target-list
   FROM  Sailors, Reserves
  [WHERE             ];
```

| Sailors | | | |
|---|---|---|---|
| sid | sname | rating | age |
| 22 | Dustin | 7 | 45.0 |
| 31 | Lubber | 8 | 55.5 |
| 58 | Rusty | 10 | 35.0 |

| Reserves | | |
|---|---|---|
| sid | bid | day |
| 22 | 101 | 10/10/96 |
| 58 | 103 | 11/12/96 |

# Find the names of sailors who have reserved boat number 103

```
SELECT [DISTINCT] target-list
   FROM Sailors, Reserves
  WHERE Sailors.sid = Reserves.sid and
        bid = 103;
```

Sailors

| sid | sname | rating | age |
|---|---|---|---|
| 22 | Dustin | 7 | 45.0 |
| 31 | Lubber | 8 | 55.5 |
| 58 | Rusty | 10 | 35.0 |

Reserves

| sid | bid | day |
|---|---|---|
| 22 | 101 | 10/10/96 |
| 58 | 103 | 11/12/96 |

# Find the names of sailors who have reserved boat number 103

```
SELECT [DISTINCT] target-list
  FROM Sailors, Reserves
 WHERE Sailors.sid = Reserves.sid and
       bid = 103;
```

| Sailors | | | |
|---|---|---|---|
| sid | sname | rating | age |
| 22 | Dustin | 7 | 45.0 |
| 31 | Lubber | 8 | 55.5 |
| 58 | Rusty | 10 | 35.0 |

| Reserves | | |
|---|---|---|
| sid | bid | day |
| 22 | 101 | 10/10/96 |
| 58 | 103 | 11/12/96 |

# Find the names of sailors who have reserved boat number 103

```
SELECT sname
  FROM Sailors, Reserves
 WHERE Sailors.sid = Reserves.sid and
       bid = 103;
```

| Sailors | | | |
|---|---|---|---|
| sid | sname | rating | age |
| 22 | Dustin | 7 | 45.0 |
| 31 | Lubber | 8 | 55.5 |
| 58 | Rusty | 10 | 35.0 |

| Reserves | | |
|---|---|---|
| sid | bid | day |
| 22 | 101 | 10/10/96 |
| 58 | 103 | 11/12/96 |

# Find the names of sailors who have reserved boat number 103

```
SELECT sname
  FROM Sailors, Reserves
 WHERE Sailors.sid = Reserves.sid and
       bid = 103;
```

How would we phrase this query in relational algebra?

# Find the names of sailors who have reserved boat number 103

```
SELECT sname
  FROM Sailors, Reserves
 WHERE Sailors.sid = Reserves.sid and
       bid = 103;
```

$$\pi_{sname}(\sigma_{bid=103}(Sailors \bowtie Reserves))$$

# Range variables

```
SELECT sname
  FROM Sailors, Reserves
 WHERE Sailors.sid = Reserves.sid and
       bid = 103;
```

```
SELECT S.sname
  FROM Sailors S, Reserves R
 WHERE S.sid = R.sid and
       R.bid = 103;
```

# What information is sought with the following query?

```
SELECT S.sname
  FROM Sailors S, Reserves R
 WHERE S.sid= R.sid;
```

Sailors

| sid | sname | rating | age |
|---|---|---|---|
| 22 | Dustin | 7 | 45.0 |
| 31 | Lubber | 8 | 55.5 |
| 58 | Rusty | 10 | 35.0 |

Reserves

| sid | bid | day |
|---|---|---|
| 22 | 101 | 10/10/96 |
| 58 | 103 | 11/12/96 |

# What information is sought in the following query?

```
SELECT S.sname
  FROM Sailors S, Reserves R
 WHERE S.sid= R.sid;
```

Find the names of the sailors who have reserved at least one boat

# What information is sought in the following query?

```
SELECT B.color
  FROM Sailors S, Reserves R, Boats B
 WHERE S.sid= R.sid and R.bid=B.bid and
 S.sname=‘Lubber’;
```

# Example Relations

| Sailors | | | |
|---|---|---|---|
| sid | sname | rating | age |
| 22 | Dustin | 7 | 45.0 |
| 31 | Lubber | 8 | 55.5 |
| 58 | Rusty | 10 | 35.0 |

| Boats | | |
|---|---|---|
| bid | bname | color |
| 101 | Interlake | blue |
| 103 | Clipper | green |

| Reserves | | |
|---|---|---|
| sid | bid | day |
| 22 | 101 | 10/10/96 |
| 58 | 103 | 11/12/96 |

# What information is sought in the following query?

```
SELECT S.color
  FROM Sailors S, Reserves R, boats B
 WHERE S.sid= R.sid and R.bid=B.bid and
 S.sname=‘Lubber’;
```

# What information is sought in the following query?

```
SELECT S.color
  FROM Sailors S, Reserves R, boats B
 WHERE S.sid= R.sid and R.bid=B.bid and
 S.sname=‘Lubber’;
```

Find the colors of boats reserved by Lubber

# Rename fields in SELECT

```
SELECT S.sname AS name
  FROM Sailors S, Reserves R
 WHERE S.sid= R.sid;
```

# Rename fields in SELECT

```
SELECT S.sname AS name
  FROM Sailors S, Reserves R
 WHERE S.sid= R.sid;
```

AS and = are two ways to name fields in result

# Rename fields in SELECT

```
SELECT name=S.sname
  FROM Sailors S, Reserves R
 WHERE S.sid= R.sid;
```

AS and = are two ways to name fields in result

# Arithmetic expressions in SELECT

```
SELECT age2=S.age*2
  FROM Sailors S, Reserves R
 WHERE S.sid= R.sid;
```

# String matching

```
SELECT s.name
  FROM Sailors S, Reserves R
 WHERE S.name LIKE ‘B_%B’;
```

# String matching

```
SELECT s.name
  FROM Sailors S, Reserves R
 WHERE S.name LIKE ‘B_%B’;
```

LIKE is used for string matching. `_’ stands for any one character and `%’ stands for 0 or more arbitrary characters

# Compute increments for the rating of persons who have sailed two different boats in the same day

```
SELECT [DISTINCT] target-list
   FROM
 WHERE condition;
```

| Sailors | | | |
|---|---|---|---|
| sid | sname | rating | age |
| 22 | Dustin | 7 | 45.0 |
| 31 | Lubber | 8 | 55.5 |
| 58 | Rusty | 10 | 35.0 |

| Reserves | | |
|---|---|---|
| sid | bid | day |
| 22 | 101 | 10/10/96 |
| 58 | 103 | 11/12/96 |

# Compute increments for the rating of persons who have sailed two different boats in the same day

```
SELECT [DISTINCT] target-list
  FROM Sailors s, Reserves R1,
       Reserves R2
 WHERE condition;
```

| Sailors | | | |
|---|---|---|---|
| sid | sname | rating | age |
| 22 | Dustin | 7 | 45.0 |
| 31 | Lubber | 8 | 55.5 |
| 58 | Rusty | 10 | 35.0 |

| Reserves | | |
|---|---|---|
| sid | bid | day |
| 22 | 101 | 10/10/96 |
| 58 | 103 | 11/12/96 |

# Compute increments for the rating of persons who have sailed two different boats in the same day

```
SELECT [DISTINCT] target-list
   FROM Sailors s, Reserves R1,
        Reserves R2
  WHERE                ;
```

| Sailors | | | |
|---|---|---|---|
| sid | sname | rating | age |
| 22 | Dustin | 7 | 45.0 |
| 31 | Lubber | 8 | 55.5 |
| 58 | Rusty | 10 | 35.0 |

| Reserves | | |
|---|---|---|
| sid | bid | day |
| 22 | 101 | 10/10/96 |
| 58 | 103 | 11/12/96 |

# Compute increments for the rating of persons who have sailed two different boats in the same day

```
SELECT [DISTINCT] target-list
  FROM Sailors s, Reserves R1,
       Reserves R2
 WHERE R1.sid=R2.sid and R1.sid=S.sid
                                      ;
```

Each record regards the same sailor

# Compute increments for the rating of persons who have sailed two different boats in the same day

```
SELECT [DISTINCT] target-list
   FROM Sailors s, Reserves R1,
        Reserves R2
  WHERE R1.sid=R2.sid and R1.sid=S.sid
        and R1.day=R2.day
                                      ;
```

And the same day

# Compute increments for the rating of persons who have sailed two different boats in the same day

```
SELECT [DISTINCT] target-list
  FROM Sailors s, Reserves R1,
       Reserves R2
 WHERE R1.sid=R2.sid and R1.sid=S.sid
       and R1.day=R2.day
       and R1.bid<>R2.bid;
```

And boat id is different in each record

# Compute increments for the rating of persons who have sailed two different boats in the same day

```
SELECT [DISTINCT] target-list
   FROM Sailors s, Reserves R1,
        Reserves R2
 WHERE R1.sid=R2.sid and R1.sid=S.sid
       and R1.day=R2.day
       and R1.bid<>R2.bid;
```

And boat id is different in each record

Check this out! https://www.w3schools.com/sql/sql_operators.asp

# Compute increments for the rating of persons who have sailed two different boats in the same day

```
SELECT [DISTINCT] target-list
  FROM Sailors s, Reserves R1,
       Reserves R2
 WHERE R1.sid=R2.sid and R1.sid=S.sid
       and R1.day=R2.day
       and R1.bid<>R2.bid;
```

# Compute increments for the rating of persons who have sailed two different boats in the same day

```
SELECTS.name, S.rating+1 AS rating
  FROM Sailors s, Reserves R1,
       Reserves R2
 WHERE R1.sid=R2.sid and R1.sid=S.sid
       and R1.day=R2.day
       and R1.bid<>R2.bid;
```

# UNION

```
SELECT DISTINCT sname
  FROM Sailors
 WHERE rating>7
UNION
SELECT DISTINCT sname
  FROM Sailors
 WHERE rating<7;
```

same as in relational algebra

# UNION

```
SELECT DISTINCT sname
   FROM Sailors
 WHERE rating>7
UNION
SELECT DISTINCT sname
   FROM Sailors
 WHERE rating<7;
```

The names of all sailors who's rating are not 7

# INTERSECT

```
SELECT DISTINCT sname
   FROM Sailors
 WHERE rating=>7
INTERSECT
SELECT DISTINCT sname
   FROM Sailors
 WHERE rating<=7;
```

same as in relational algebra

# INTERSECT

```
SELECT DISTINCT sname
   FROM Sailors
 WHERE rating=>7
INTERSECT
SELECT DISTINCT sname
   FROM Sailors
 WHERE rating<=7;
```

The names of all sailors who's rating equals 7

# EXCEPT (MINUS)

```
SELECT DISTINCT sname
  FROM Sailors
 WHERE rating>7
EXCEPT
SELECT DISTINCT sname
  FROM Sailors
 WHERE rating>9;
```

Same as set difference from relational algebra

# EXCEPT (MINUS)

```
SELECT DISTINCT sname
   FROM Sailors
 WHERE rating>7
EXCEPT
SELECT DISTINCT sname
   FROM Sailors
 WHERE rating>9;
```

The names of all sailors who's rating equals 8 or 9

# ORDER BY

- Used to sort results by one or more columns
- Default sorting: in ascending order
- Specify ASC or DESC if needed

# ORDER BY

```
SELECT sname, rating, age
  FROM Sailors S
 WHERE age>17
ORDER BY rating ASC, age DESC
```

- Primary ascending sort by rating
- Secondary descending sort by age

# Nested Queries

A very powerful feature of SQL

# Nested Queries

Find names of sailors who’ve reserved boat #103:

# Find the names of sailors who have reserved boat number 103

```
SELECT sname
  FROM Sailors, Reserves
 WHERE Sailors.sid = Reserves.sid and
       bid = 103;
```

$$\pi_{sname}(\sigma_{bid=103}(Sailors \bowtie Reserves))$$

# Nested Queries

Find names of sailors who’ve reserved boat #103:

```
SELECT  R.sid
  FROM  Reserves R
 WHERE  R.bid=103
```

# Nested Queries

Find names of sailors who’ve reserved boat #103:

```
(SELECT  R.sid
   FROM  Reserves R
  WHERE  R.bid=103)
```

# Nested Queries

Find names of sailors who’ve reserved boat #103:

```
SELECT  S.sname
  FROM  Sailors S
 WHERE  S.sid IN (SELECT  R.sid
                    FROM  Reserves R
                   WHERE  R.bid=103)
```

# Nested Queries

A WHERE clause can itself contain an SQL query!

```
SELECT  S.sname
  FROM  Sailors S
 WHERE  S.sid IN (SELECT  R.sid
                    FROM  Reserves R
                   WHERE  R.bid=103)
```

# Nested Queries

To find sailors who’ve *not* reserved #103, use NOT  IN

```
SELECT  S.sname
  FROM  Sailors S
 WHERE  S.sid IN (SELECT  R.sid
                    FROM  Reserves R
                   WHERE  R.bid=103)
```

# You can nest as much as you want

```
SELECT  S.sname
  FROM  Sailors S
 WHERE  S.sid NOT IN
            (SELECT  R.sid
               FROM  Reserves R
              WHERE  R.bid IN
                      (SELECT  B.bid
                         FROM  Boats B
                        WHERE  B.color=‘red’))
```

# Output: set of bid of boats that are red

```
SELECT  B.bid
  FROM  Boats B
 WHERE  B.color='red'
```

# A set of reservation id’s for red boats

```
(SELECT  R.sid
   FROM  Reserves R
  WHERE  R.bid IN
          (SELECT  B.bid
             FROM  Boats B
            WHERE  B.color=‘red’))
```

# Name of sailor who have **not** reserved a red boat

```
SELECT  S.sname
  FROM  Sailors S
 WHERE  S.sid NOT IN
           (SELECT  R.sid
              FROM  Reserves R
             WHERE  R.bid IN
                      (SELECT  B.bid
                         FROM  Boats B
                        WHERE  B.color=‘red’))
```

# Set comparison operators

Find the sailor with the highest rating

# Example Relations

| Sailors | | | |
|---|---|---|---|
| sid | sname | rating | age |
| 22 | Dustin | 7 | 45.0 |
| 31 | Lubber | 8 | 55.5 |
| 58 | Rusty | 10 | 35.0 |

| Boats | | |
|---|---|---|
| bid | bname | color |
| 101 | Interlake | blue |
| 103 | Clipper | green |

| Reserves | | |
|---|---|---|
| sid | bid | day |
| 22 | 101 | 10/10/96 |
| 58 | 103 | 11/12/96 |

# Set comparison operators

Find the sailor with the highest rating

```
SELECT  S.sid
  FROM  Sailors S
 WHERE  S.rating >= ALL
                    (SELECT  S2.rating
                       FROM  Sailors S2)
```

# Set comparison operators

Find the sailor with the highest rating

```
SELECT  S.sid
  FROM  Sailors S
 WHERE  S.rating >= ALL
                    (SELECT  S2.rating
                       FROM  Sailors S2)
```

# Set comparison operators

In the same way we can use ANY

(as well as combine with other operators)

```
SELECT  S.sid
  FROM  Sailors S
 WHERE  S.rating >= ALL
                   (SELECT  S2.rating
                      FROM  Sailors S2)
```

# Correlated Nested Queries

So far, subquery was independent from query-

**that is not always the case**

# Correlated Nested Queries

```
SELECT  S.sname
  FROM  Sailors S
 WHERE EXISTS (SELECT  *
                 FROM  Reserves R
                WHERE  R.bid=103 and
                       S.sid=R.sid)
```

# Correlated Nested Queries

```
SELECT  S.sname
  FROM  Sailors S
 WHERE EXISTS (SELECT  *
                 FROM  Reserves R
                WHERE  R.bid=103 and
                       S.sid=R.sid)
```

EXISTS is an operator that checks if the output from the inner query contains records (or empty)

# Correlated Nested Queries

```
SELECT  S.sname
  FROM  Sailors S
 WHERE EXISTS (SELECT  *
                 FROM  Reserves R
                WHERE  R.bid=103 and
                       S.sid=R.sid)
```

# Correlated Nested Queries

```
SELECT  S.sname
  FROM  Sailors S
 WHERE EXISTS (SELECT  *
                 FROM  Reserves R
                WHERE  R.bid=103 and
                       S.sid=R.sid)
```

Checks for each record in Sailors

# Correlated Nested Queries

```
SELECT  S.sname
  FROM  Sailors S
 WHERE NOT EXISTS (SELECT  *
                    FROM  Reserves R
                   WHERE  R.bid=103 and
                          S.sid=R.sid)
```

We can also use NOT EXIST

# Division- reminder from relational algebra: employees who have passed all classes in B2

| eno | cno |
|---|---|
| S1 | P1 |
| S1 | P2 |
| S1 | P3 |
| S1 | P4 |
| S2 | P1 |
| S2 | P2 |
| S3 | P2 |
| S4 | P2 |
| S4 | P4 |

A

/

| cno |
|---|
| P2 |
| P4 |

B2

=

| eno |
|---|
| S1 |
| S4 |

In SQL: not as easy to express

Alternative phrasing – “all employees that **have not failed any** class in B”

# With sailors:

From:

“Names of sailors who have reserved all boats”

To:

“Names of sailors for which there is no boat that they did not reserve”

# Division

Names of sailors for which there is no boat that they did not reserve

```
SELECT  S.sname
  FROM  Sailors S
 WHERE NOT EXISTS((SELECT  B.bid
                     FROM  Boats B)
                   EXCEPT
                  (SELECT  R.bid
                     FROM  Reserves R
                    WHERE  S.sid=R.sid))
```

# Division

Names of sailors for which there is no boat that they did not reserve

```
SELECT  S.sname
  FROM  Sailors S
 WHERE NOT EXISTS((SELECT  B.bid
                    FROM  Boats B)
                  EXCEPT
                 (SELECT  R.bid
                    FROM  Reserves R
                   WHERE  S.sid=R.sid))
```

# Division

Names of sailors for which there is no boat that they did not reserve

```
SELECT  S.sname
  FROM  Sailors S
 WHERE NOT EXISTS((SELECT  B.bid
                     FROM  Boats B)
                   EXCEPT
                  (SELECT  R.bid
                     FROM  Reserves R
                    WHERE  S.sid=R.sid))
```

# Division

Names of sailors for which there is no boat that they did not reserve

```
SELECT  S.sname
  FROM  Sailors S
 WHERE NOT EXISTS((SELECT  B.bid
                     FROM  Boats B)
                   EXCEPT
                  (SELECT  R.bid
                     FROM  Reserves R
                    WHERE  S.sid=R.sid))
```

# Division- without EXCEPT

Names of sailors that there is no boat to which reservation in their name was not made

```
SELECT  S.sname
  FROM  Sailors S
 WHERE NOT EXISTS(SELECT  B.bid
                    FROM  Boats B
     WHERE NOT EXISTS(SELECT  R.bid
                        FROM  Reserves R
                       WHERE  R.bid=B.bid
                          and S.sid=R.sid))
```

# Division- without EXCEPT

Names of sailors that there is no boat to which reservation in their name was not made

```
SELECT  S.sname
  FROM  Sailors S
 WHERE NOT EXISTS(SELECT  B.bid
                    FROM  Boats B
     WHERE NOT EXISTS(SELECT  R.bid
                        FROM  Reserves R
                       WHERE  R.bid=B.bid
                          and S.sid=R.sid))
```

# Division- without EXCEPT

Names of sailors that there is no boat to which reservation in their name was not made

```
SELECT  S.sname
  FROM  Sailors S
 WHERE NOT EXISTS(SELECT  B.bid
                    FROM  Boats B
     WHERE NOT EXISTS(SELECT  R.bid
                        FROM  Reserves R
                       WHERE  R.bid=B.bid
                         and  S.sid=R.sid))
```

# Division- without EXCEPT

Names of sailors that there is no boat to which reservation in their name was not made

```
SELECT  S.sname
  FROM  Sailors S
 WHERE NOT EXISTS(SELECT  B.bid
                    FROM  Boats B
     WHERE NOT EXISTS(SELECT  R.bid
                        FROM  Reserves R
                       WHERE  R.bid=B.bid
                          and S.sid=R.sid))
```

# Aggregate

# Aggregate Queries

Find the average age of sailors with rating of 10

# Aggregate Queries

Find the average age of sailors with rating of 10

```
SELECT AVG(S.age)
  FROM Sailors S
 WHERE S.rating=10
```

# Aggregate Operators

- `AVG([DISTINCT] A)`
- `SUM([DISTINCT] A)`
- `MAX(A)`
- `MIN(A)`
- `COUNT([DISTINCT] A)`
- `ORDER`

# Aggregate Queries

Sum the ages of sailors with rating of 10

```
SELECT  SUM(S.age)
  FROM Sailors S
 WHERE S.rating=10
```

# Aggregate Queries

Find the youngest sailor that has a rating of 10

```
SELECT MIN(S.age)
  FROM Sailors S
 WHERE S.rating=10
```

# Aggregate Queries

Find the oldest sailor that has a rating of 10

```
SELECT MAX(S.age)
  FROM Sailors S
 WHERE S.rating=10
```

# Aggregate Queries

Counts all sailors

```
SELECT COUNT(*)
  FROM Sailors S
```

# Aggregate Queries

Counts all sailors that have a rating of 10

```
SELECT COUNT(S.sid)
  FROM Sailors S
 WHERE S.rating=10
```

# Aggregate Queries

Counts all different names of the sailors

```
SELECT COUNT(DISTINCT S.name)
  FROM Sailors S
```

# GROUP BY and HAVING

# Motivation

```
Find the age of the youngest sailor for
          each rating level
```

In general, we don't know how many rating levels exist, and what is the rating for each level

# Basic SQL Query

```
SELECT [DISTINCT] target-list
   FROM relation-list
  [WHERE condition];
```

- `relation-list`: list of relation names
- `target-list`: list of attributes onto which the output relation is projected
- `Condition`: optional Boolean condition

# GROUP BY SQL Query

```
SELECT [DISTINCT] target-list
  FROM relation-list
 WHERE condition
GROUP BY grouping-list
[HAVING group-condition];
```

- *grouping-list*: list of attributes
- *group-condition* : a Boolean condition

# GROUP BY SQL Query

```
SELECT [DISTINCT] target-list
   FROM relation-list
  WHERE condition
GROUP BY grouping-list
[HAVING group-condition];
```

- *target-list*:

  (i) names of attribute  from the grouping-list

  (ii) terms with aggregate operations (e.g., MIN (S.age))

# Conceptual evaluation:

```
SELECT [DISTINCT] target-list
  FROM relation-list
 WHERE condition
GROUP BY grouping-list
[HAVING group-condition];
```

1. Compute the cross-product of *relation-list*
2. Discard resulting tuples if they fail *condition*
3. Remaining tuples are partitioned into groups by the value of attributes in *grouping-list*
4. The *group-qualification* is then applied to eliminate some groups.  Expressions in *group-qualification* must have a single value per group!
5. One answer tuple is generated per qualifying group according to the *target-list*
6. If DISTINCT is specified, eliminate duplicate rows

Find the age of the youngest sailor for each rating level

```
SELECT S.rating, MIN(S.age)
  FROM Sailors S
GROUP BY S.rating;
```

```
  SELECT S.rating, MIN(S.age)
    FROM Sailors S
GROUP BY S.rating;
```

| Sailors | | | |
|---|---|---|---|
| sid | sname | rating | age |
| 22 | Dustin | 7 | 45 |
| 31 | Lubber | 8 | 55.5 |
| 58 | Rusty | 10 | 35 |
| 64 | Horatio | 7 | 35 |
| 29 | Brutus | 1 | 33 |
| 71 | Zorba | 10 | 16 |

```
  SELECT S.rating, MIN(S.age)
    FROM Sailors S
GROUP BY S.rating;
```

| Sailors | | | |
|---|---|---|---|
| sid | sname | rating | age |
| 22 | Dustin | 7 | 45 |
| 31 | Lubber | 8 | 55.5 |
| 58 | Rusty | 10 | 35 |
| 64 | Horatio | 7 | 35 |
| 29 | Brutus | 1 | 33 |
| 71 | Zorba | 10 | 16 |

```
  SELECT S.rating, MIN(S.age)
    FROM Sailors S
GROUP BY S.rating;
```

| Sailors | | | |
|---|---|---|---|
| sid | sname | rating | age |
| 22 | Dustin | 7 | 45 |
| 31 | Lubber | 8 | 55.5 |
| 58 | Rusty | 10 | 35 |
| 64 | Horatio | 7 | 35 |
| 29 | Brutus | 1 | 33 |
| 71 | Zorba | 10 | 16 |

```
   SELECT S.rating, MIN(S.age)
     FROM Sailors S
GROUP BY S.rating;
```

| Sailors | | | |
|---|---|---|---|
| sid | sname | rating | age |
| 22 | Dustin | 7 | 45 |
| 31 | Lubber | 8 | 55.5 |
| 58 | Rusty | 10 | 35 |
| 64 | Horatio | 7 | 35 |
| 29 | Brutus | 1 | 33 |
| 71 | Zorba | 10 | 16 |

```
   SELECT S.rating, MIN(S.age)
     FROM Sailors S
GROUP BY S.rating;
```

| Sailors | | | |
|---|---|---|---|
| sid | sname | rating | age |
| 31 | Lubber | 8 | 55.5 |

| Sailors | | | |
|---|---|---|---|
| sid | sname | rating | age |
| 29 | Brutus | 1 | 33 |

| Sailors | | | |
|---|---|---|---|
| sid | sname | rating | age |
| 58 | Rusty | 10 | 35 |
| 71 | Zorba | 10 | 16 |

| Sailors | | | |
|---|---|---|---|
| sid | sname | rating | age |
| 22 | Dustin | 7 | 45 |
| 64 | Horatio | 7 | 35 |

```
  SELECT S.rating, MIN(S.age)
    FROM Sailors S
GROUP BY S.rating;
```

| Sailors | | | |
|---|---|---|---|
| sid | sname | rating | age |
| 31 | Lubber | 8 | 55.5 |

| Sailors | | | |
|---|---|---|---|
| sid | sname | rating | age |
| 29 | Brutus | 1 | 33 |

| Sailors | | | |
|---|---|---|---|
| sid | sname | rating | age |
| 58 | Rusty | 10 | 35 |
| 71 | Zorba | 10 | 16 |

| Sailors | | | |
|---|---|---|---|
| sid | sname | rating | age |
| 22 | Dustin | 7 | 45 |
| 64 | Horatio | 7 | 35 |

```
  SELECT S.rating, MIN(S.age)
    FROM Sailors S
GROUP BY S.rating;
```

Sailors

| sid | sname | rating | age |
|---|---|---|---|
| 31 | Lubber | 8 | 55.5 |

Sailors

| sid | sname | rating | age |
|---|---|---|---|
| 29 | Brutus | 1 | 33 |

Sailors

| sid | sname | rating | age |
|---|---|---|---|
| 58 | Rusty | 10 | 35 |
| 71 | Zorba | 10 | 16 |

Sailors

| sid | sname | rating | age |
|---|---|---|---|
| 22 | Dustin | 7 | 45 |
| 64 | Horatio | 7 | 35 |

```
  SELECT S.rating, MIN(S.age)
    FROM Sailors S
GROUP BY S.rating;
```

Sailors

| sid | sname | rating | age |
|---|---|---|---|
| 31 | Lubber | 8 | 55.5 |

Sailors

| sid | sname | rating | age |
|---|---|---|---|
| 29 | Brutus | 1 | 33 |

Sailors

| sid | sname | rating | age |
|---|---|---|---|
| 58 | Rusty | 10 | 35 |
| 71 | Zorba | 10 | 16 |

Sailors

| sid | sname | rating | age |
|---|---|---|---|
| 22 | Dustin | 7 | 45 |
| 64 | Horatio | 7 | 35 |

```sql
  SELECT S.rating, MIN(S.age)
    FROM Sailors S
GROUP BY S.rating;
```

Rating: 8
Minimum age: 55.5

Rating: 1
Minimum age: 33

Rating: 10
Minimum age: 16

Rating: 7
Minimum age: 35

```
  SELECT S.rating, MIN(S.age)
  FROM Sailors S
GROUP BY S.rating;
```

| rating | age |
|---|---|
| 8 | 55.5 |
| 1 | 33 |
| 10 | 16 |
| 7 | 35 |

## Find the age of the youngest sailor for each rating level who is eligible to vote

```
SELECT S.rating, MIN(S.age)
  FROM Sailors S
 WHERE S.age >= 18
GROUP BY S.rating
```

```
SELECT S.rating, MIN(S.age)
  FROM Sailors S
 WHERE S.age >= 18
GROUP BY S.rating
```

| Sailors | | | |
|---|---|---|---|
| sid | sname | rating | age |
| 22 | Dustin | 7 | 45 |
| 31 | Lubber | 8 | 55.5 |
| 58 | Rusty | 10 | 35 |
| 64 | Horatio | 7 | 35 |
| 29 | Brutus | 1 | 33 |
| ~~71~~ | ~~Zorba~~ | ~~10~~ | ~~16~~ |

```
SELECT S.rating, MIN(S.age)
  FROM Sailors S
 WHERE S.age >= 18
GROUP BY S.rating
```

| rating | age |
|---|---|
| 8 | 55.5 |
| 1 | 33 |
| 10 | 35 |
| 7 | 35 |

For each rating level that have at least two sailors, return the rating and the age of the youngest sailor

```
SELECT S.rating, MIN(S.age)
  FROM Sailors S
GROUP BY S.rating
HAVING count(*)>1;
```

```
  SELECT S.rating, MIN(S.age)
    FROM Sailors S
GROUP BY S.rating
HAVING count(*)>1;
```

| Sailors | | | |
|---|---|---|---|
| sid | sname | rating | age |
| 22 | Dustin | 7 | 45 |
| 31 | Lubber | 8 | 55.5 |
| 58 | Rusty | 10 | 35 |
| 64 | Horatio | 7 | 35 |
| 29 | Brutus | 1 | 33 |
| 71 | Zorba | 10 | 16 |

```
  SELECT S.rating, MIN(S.age)
    FROM Sailors S
GROUP BY S.rating
HAVING COUNT(*)>1;
```

Sailors

| sid | sname | rating | age |
|---|---|---|---|
| 31 | Lubber | 8 | 55.5 |

Sailors

| sid | sname | rating | age |
|---|---|---|---|
| 29 | Brutus | 1 | 33 |

Sailors

| sid | sname | rating | age |
|---|---|---|---|
| 58 | Rusty | 10 | 35 |
| 71 | Zorba | 10 | 16 |

Sailors

| sid | sname | rating | age |
|---|---|---|---|
| 22 | Dustin | 7 | 45 |
| 64 | Horatio | 7 | 35 |

```
  SELECT S.rating, MIN(S.age)
    FROM Sailors S
GROUP BY S.rating
HAVING COUNT(*)>1;
```

| Sailors | | | |
|---|---|---|---|
| sid | sname | rating | age |
| 31 | Lubber | 8 | 55.5 |

| Sailors | | | |
|---|---|---|---|
| sid | sname | rating | age |
| 29 | Brutus | 1 | 33 |

| Sailors | | | |
|---|---|---|---|
| sid | sname | rating | age |
| 58 | Rusty | 10 | 35 |
| 71 | Zorba | 10 | 16 |

| Sailors | | | |
|---|---|---|---|
| sid | sname | rating | age |
| 22 | Dustin | 7 | 45 |
| 64 | Horatio | 7 | 35 |

```
  SELECT S.rating, MIN(S.age)
    FROM Sailors S
GROUP BY S.rating
HAVING COUNT(*)>1;
```

| Sailors | | | |
|---|---|---|---|
| sid | sname | rating | age |
| 31 | Lubber | 8 | 55.5 |

| Sailors | | | |
|---|---|---|---|
| sid | sname | rating | age |
| 29 | Brutus | 1 | 33 |

| Sailors | | | |
|---|---|---|---|
| sid | sname | rating | age |
| 58 | Rusty | 10 | 35 |
| 71 | Zorba | 10 | 16 |

| Sailors | | | |
|---|---|---|---|
| sid | sname | rating | age |
| 22 | Dustin | 7 | 45 |
| 64 | Horatio | 7 | 35 |

```
  SELECT S.rating, MIN(S.age)
    FROM Sailors S
GROUP BY S.rating
HAVING COUNT(*)>1;
```

| Sailors | | | |
|---|---|---|---|
| sid | sname | rating | age |
| 58 | Rusty | 10 | 35 |
| 71 | Zorba | 10 | 16 |

| Sailors | | | |
|---|---|---|---|
| sid | sname | rating | age |
| 22 | Dustin | 7 | 45 |
| 64 | Horatio | 7 | 35 |

```
  SELECT S.rating, MIN(S.age)
    FROM Sailors S
GROUP BY S.rating
HAVING COUNT(*)>1;
```

Rating: 10
Minimum age: 16

Rating: 7
Minimum age: 35

```
SELECT S.rating, MIN(S.age)
  FROM Sailors S
GROUP BY S.rating
HAVING count(*)>1;
```

| rating | age |
|---|---|
| 10 | 16 |
| 7 | 35 |

# Nested query in SELECT

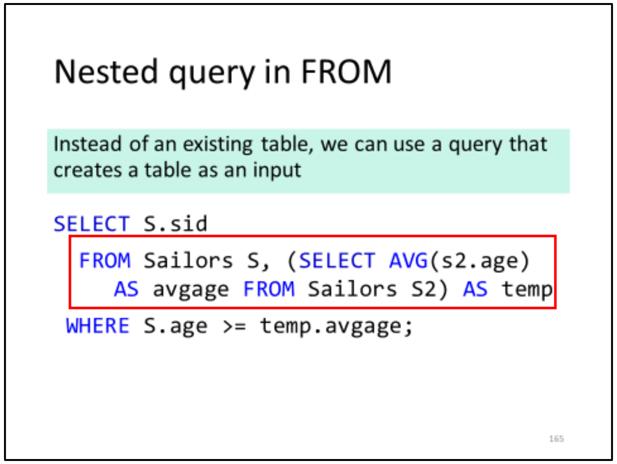
We can nest a query in the select clause, but it has to return at most one value for each record that is returned by the outer query

```
SELECT S.sid, S.age,(SELECT AVG(S2.age)
                       FROM Sailors S2)
  FROM Sailors s;
```

# Nested query in FROM

Instead of an existing table, we can use a query that creates a table as an input

```
SELECT S.sid
  FROM Sailors S, (SELECT AVG(s2.age)
     AS avgage FROM Sailors S2) AS temp
 WHERE S.age >= temp.avgage;
```



## Nested query in FROM

Instead of an existing table, we can use a query that creates a table as an input

```
SELECT S.sid
  FROM Sailors S, (SELECT AVG(s2.age)
     AS avgage FROM Sailors S2) AS temp
 WHERE S.age >= temp.avgage;
```

```
FROM Sailors S,
    (SELECT AVG(s2.age) AS avgage
    FROM Sailors S2) AS temp
```

# Nested query in FROM

Instead of an existing table, we can use a query that creates a table as an input

```
SELECT S.sid
  FROM Sailors S, (SELECT AVG(s2.age)
     AS avgage FROM Sailors S2) AS temp
 WHERE S.age >= temp.avgage;
```

# Null values

- Expressions that involve null (e.g. null+3) -> null
- In logical expressions null is equal to false:
  - (Null AND 1) -> false
- In aggregation functions:
  - COUNT(*) –counts all tuples including null values
  - COUNT(R) – counts only non-null records
  - SUM,AVG,MIN,MAX – ignore null values (if all values are null the result will be null)
- Records are considered identical if they have matching identical non-null values
  - DISTINCT eliminates identical records from the result
  - GROUP BY groups according to identical records

# Assertion constraints

IC over several tables

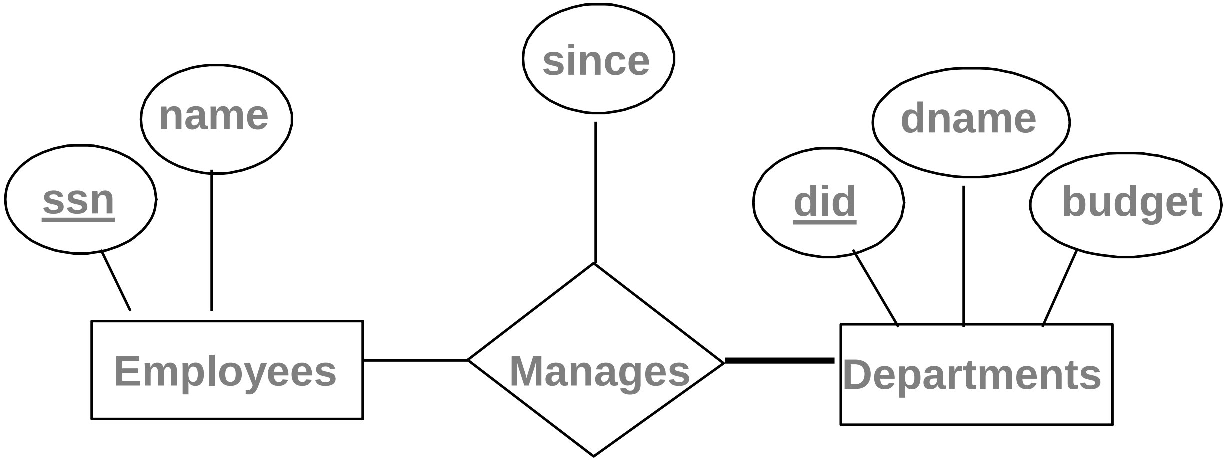
# Assertion constraints

- A more sophisticated form of check constrains
- Enables enforcement of conditions that involves several tables

**reminder**

# Relationship sets to tables: participation constraints

- A department is managed by at least one employee
- An employee can manage any number of departments (including none)



# Assertion constraints

```
CREATE ASSERTION EmployeesInDepts
  CHECK(
    NOT EXISTS(
      SELECT * FROM Department D
       WHERE NOT EXISTS(
         SELECT * FROM Manages M
         WHERE D.did == M.did ) ) )
```

- Employees(ssn, name, title)
- manages(since, did, ssn)
- Departments(did, dname, budget)

# Assertion constraints

```
CREATE ASSERTION EmployeesInDepts
  CHECK(
    NOT EXISTS(
      SELECT * FROM Department D
       WHERE NOT EXISTS(
         SELECT * FROM Manages M
         WHERE D.did == M.did ) ) )
```

- Employees(ssn, name, title)
- manages(since, did, ssn)
- Departments(did, dname, budget)

# Assertion constraints

```
CREATE ASSERTION EmployeesInDepts
   CHECK(
      NOT EXISTS(
         SELECT * FROM Department D
          WHERE NOT EXISTS(
             SELECT * FROM Manages M
             WHERE D.did == M.did ) ) )
```

- Employees(ssn, name, title)
- manages(since, did, ssn)
- Departments(did, dname, budget)

# Assertion constraints

```
CREATE ASSERTION EmployeesInDepts
   CHECK(
      NOT EXISTS(
         SELECT * FROM Department D
          WHERE NOT EXISTS(
              SELECT * FROM Manages M
              WHERE D.did == M.did ) ) )
```

- Employees(ssn, name, title)
- manages(since, did, ssn)
- Departments(did, dname, budget)