Find those ratings for which the average age is the minimum over all ratings

- Aggregate operations cannot be nested! WRONG:

```
SELECT S.rating
FROM Sailors S
WHERE S.age = (SELECT MIN (AVG (S2.age))
                FROM Sailors S2)
```

- Correct solution (in SQL/92):

```
SELECT Temp.rating
FROM (SELECT S.rating, AVG (S.age) AS avgage
      FROM Sailors S
      GROUP BY S.rating) AS Temp
WHERE Temp.avgage = (SELECT MIN (Temp.avgage)
                     FROM Temp)
```

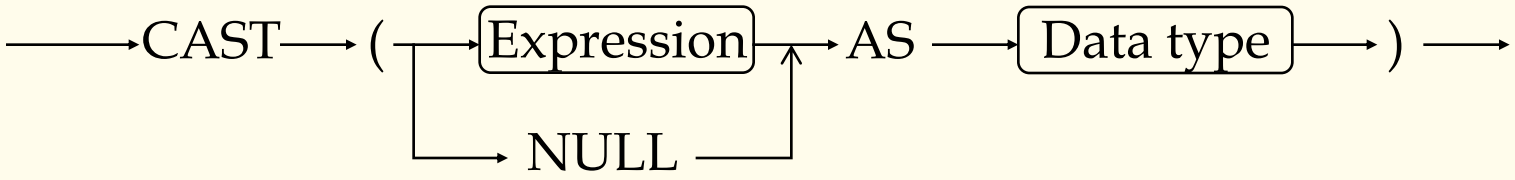
Null Values

- Field values in a tuple are sometimes *unknown* (e.g., a rating has not been assigned) or *inapplicable* (e.g., no spouse's name).
  - SQL provides a special value *null* for such situations.
- The presence of *null* complicates many issues. E.g.:
  - Special operators needed to check if value is/is not *null*.
  - Is *rating*>*8* true or false when *rating* is equal to *null*? What about AND, OR and NOT connectives?
  - We need a 3-valued logic (true, false and *unknown*).
  - Meaning of constructs must be defined carefully. (e.g., WHERE clause eliminates rows that don't evaluate to true.)
  - New operators (in particular, *outer joins*) possible/needed.

Some New Features of SQL

- **CAST expression**
- CASE expression
- Sub-query
- Outer Join
- Recursion

CAST Expression



- Change the expression to the target data type
- Valid target type
- Use
  - Match function parameters

    substr(string1, CAST(x AS Integer), CAST(y AS Integer))
  - Change precision while calculating

    CAST (elevation AS Decimal (5,0))
  - Assign a data type to NULL value

CAST Expression

- Example:

  Students (name, school)

  Soldiers (name, service)

```
CREATE VIEW prospects (name, school, service) AS
        SELECT name, school, CAST(NULL AS Varchar(20))
        FROM Students
UNION
        SELECT name, CAST(NULL AS Varchar(20)), service
        FROM Soldiers ;
```

Some New Features of SQL

- CAST expression
- **CASE expression**
- Sub-query
- Outer Join
- Recursion

CASE Expression

- Simple form :

  Officers (name, status, rank, title)

```
SELECT name, CASE status
                    WHEN 1 THEN 'Active Duty'
                    WHEN 2 THEN 'Reserve'
                    WHEN 3 THEN 'Special Assignment'
                    WHEN 4 THEN 'Retired'
                    ELSE 'Unknown'
               END AS status
FROM Officers ;
```

CASE Expression

- General form (use searching condition):

  Machines (serialno, type, year, hours_used, accidents)

- *Find the rate of the accidents of "chain saw" in the whole accidents :*

```
SELECT sum (CASE
                    WHEN type='chain saw' THEN accidents
                    ELSE 0e0
                END) / sum (accidents)
FROM Machines;
```

CASE Expression

- *Find the average accident rate of every kind of equipment :*

```
SELECT type, CASE
                        WHEN sum(hours_used)>0 THEN
                                sum(accidents)/sum(hours_used)
                        ELSE NULL
                END AS accident_rate
FROM Machines
GROUP BY type;
```

(Because some equipments maybe not in use at all, their hours_used is 0. Use CASE can prevent the expression divided by 0.)

CASE Expression

- Compared with

```
SELECT type, sum(accidents)/sum(hours_used)
FROM Machines
GROUP BY type
HAVING sum(hours_used)>0;
```