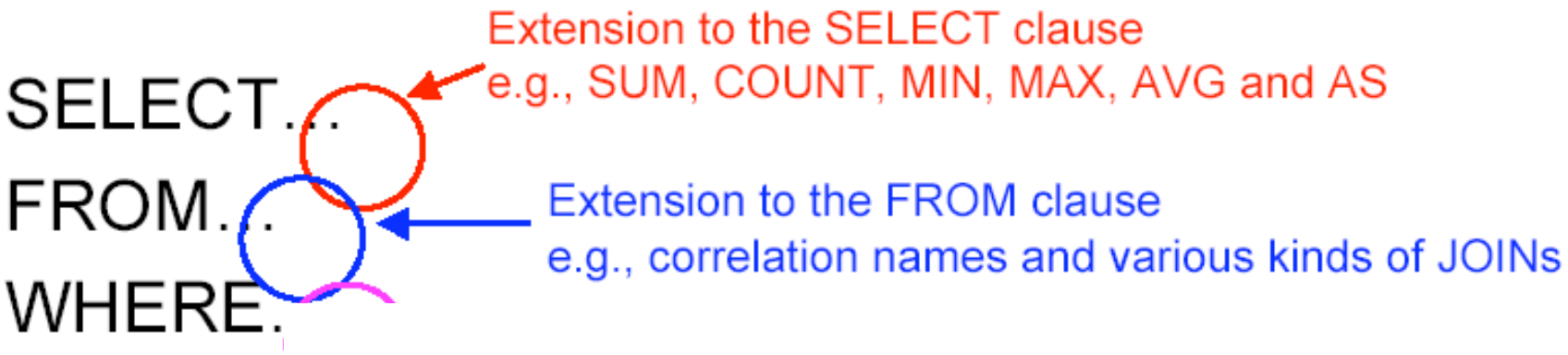
Review of Last Lecture



In-Class Exercise #1

- Find the names of employees who work for some of the projects controlled by the research department

EMPLOYEE	FNAME	MINIT	LNAME	SSN	BDATE	ADDRESS	SEX	SALARY	SUPERSSN	DNO
	John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
	Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
	Alicia	J	Zelaya	999887777	1968-07-19	3321 Castle, Spring, TX	F	25000	987654321	4
	Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
	Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
	Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
	Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
	James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	null	1

DEPT_LOCATIONS	DNUMBER	DLOCATION
	1	Houston
	4	Stafford
	5	Bellaire
	5	Sugarland
	5	Houston

DEPARTMENT	DNAME	DNUMBER	MGRSSN	MGRSTARTDATE
	Research	5	333445555	1988-05-22
	Administration	4	987654321	1995-01-01
	Headquarters	1	888665555	1981-06-19

WORKS_ON	ESSN	PNO	HOURS
	123456789	1	32.5
	123456789	2	7.5
	666884444	3	40.0
	453453453	1	20.0
	453453453	2	20.0
	333445555	2	10.0
	333445555	3	10.0
	333445555	10	10.0
	333445555	20	10.0
	999887777	30	30.0
	999887777	10	10.0
	987987987	10	35.0
	987987987	30	5.0
	987654321	30	20.0
	987654321	20	15.0
	888665555	20	null

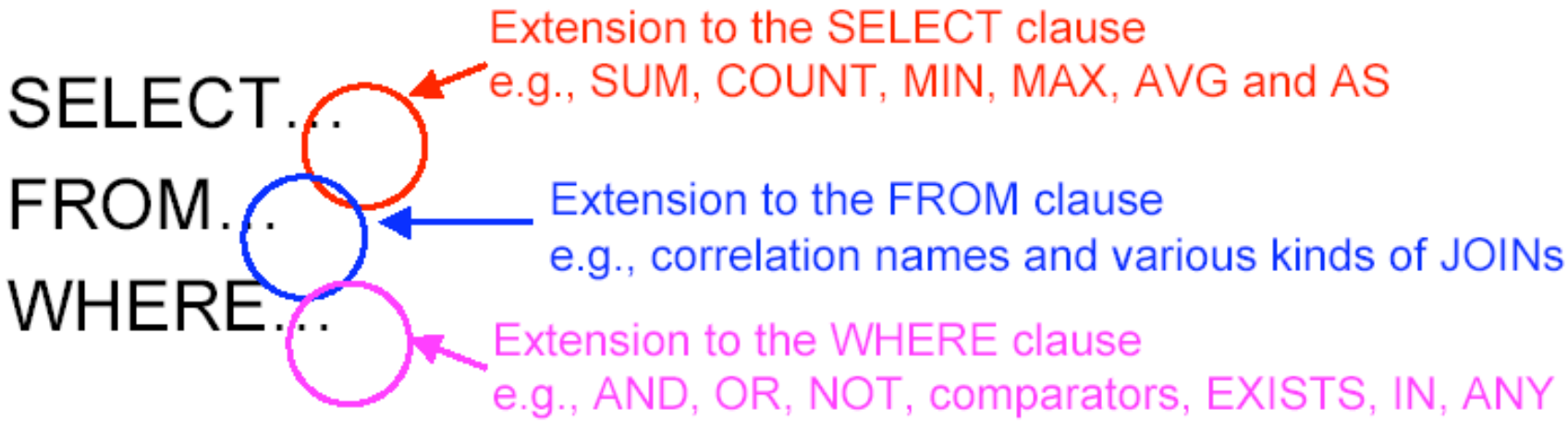
PROJECT	PNAME	PNUMBER	PLOCATION	DNUM
	ProductX	1	Bellaire	5
	ProductY	2	Sugarland	5
	ProductZ	3	Houston	5
	Computerization	10	Stafford	4
	Reorganization	20	Houston	1
	Newbenefits	30	Stafford	4

DEPENDENT	ESSN	DEPENDENT_NAME	SEX	BDATE	RELATIONSHIP
	333445555	Alice	F	1986-04-05	DAUGHTER
	333445555	Theodore	M	1983-10-25	SON
	333445555	Joy	F	1958-05-03	SPOUSE

```
SELECT   FName, LName
FROM     Employee, Department,
         Works_ON, Project
WHERE
```

Today's Plan

- Extension to Where
  - Boolean and comparator
  - Subquery
- SQL exercises



Comparator

- <, >, =, <>, >=, <=
  - Compare two values as expected
  - Operates on numbers as well as text values
- LIKE
  - Compare a text value with a pattern
  - '%' compares with zero or more characters
  - '_' compares with exactly one character

Patterns

- WHERE clauses can have conditions in which a string is compared with a pattern, to see if it matches.
- General form:

  <Attribute> LIKE <pattern> or

  <Attribute> NOT LIKE <pattern>
- Pattern is a quoted string with % = “any string”; _ = “any single character.”

```
Select   *
From     Employee
Where    LName LIKE "L%“;
```

```
Select   *
From     Employee
Where    LName LIKE "L_";
```

Pattern: Example

- From Drinkers(name, addr, phone) find the drinkers with exchange 555:

```
SELECT   name
FROM     Drinkers
WHERE    phone LIKE "%555-_ _ _ _";
```

Match?

1. (555)123 5567
2. 555-34344
3. (1)555-1134
4. 5555-66BB

Review: Nested Query in FROM Clause

```
SELECT    E.FName, E.LName
FROM      Employee E,
          (SELECT    MIN(Salary) AS MinS
           FROM      Employee) AS M
WHERE     E.Salary = M.MinS
```

Find employees with the lowest salary

Subquery in WHERE Clause

```
SELECT    E.FName, E.LName
FROM      Employee AS E
WHERE     E.Salary IN
              (SELECT    MIN(Salary)
               FROM      Employee)
```

Find employees with the lowest salary

You can think of a loop, where each employee tuple is checked against the qualification by computing the subquery

IN and NOT IN

- Use to compare a single value with a table of values
  - This table must have only one attribute and the domain must be the same as well
  - Typically associated with a subquery in the WHERE clause

```
SELECT    E.FName, E.LName
FROM      Employee AS E
WHERE     E.Salary IN
              (SELECT    MIN(Salary)
               FROM      Employee)
```

Subquery: Exercise

- Retrieve the name and address of all employees who work for the 'Research' department.

```
SELECT    E.FName, E.LName, E.Address
FROM      Employee AS E
WHERE     E.DNO IN
```

EXPLICIT SETS

- It is also possible to use an **explicit (enumerated) set of values** in the WHERE-clause rather than a nested query
- Retrieve the social security numbers of all employees who work on project number 1, 2, or 3.

```
SELECT   DISTINCT   ESSN
FROM                WORKS_ON
WHERE               PNO IN  (1, 2, 3)
```

Correlated Subquery: Example

- Retrieve the name of each employee who has a dependent with the same first name as the employee.

```
SELECT    E.FNAME, E.LNAME
FROM      EMPLOYEE AS E
WHERE     E.SSN IN
                (SELECT    ESSN
                FROM       DEPENDENT
                WHERE      ESSN=E.SSN AND
                           E.FNAME=DEPENDENT_NAME)
```

Correlated: because the subquery references two attributes from a table in the outer query

Note on Correlated Subquery

- It is very EXPENSIVE!
- For every tuple in the outer query, the correlated subquery must be evaluated once!

Non-Correlated Subquery: Example

```
SELECT    E.FName, E.LName
FROM      Employee AS E
WHERE     E.Salary IN
              (SELECT    MIN(Salary)
               FROM      Employee)
```

Find employees with the lowest salary

The subquery only uses attributes from its own table (table mentioned in the subquery)

Exists Clause

```
SELECT    E.FName, E.LName
FROM      Employee AS E
WHERE     EXISTS
                (SELECT    ESSN
                 FROM      Works_on
                 WHERE     E.SSN = ESSN);
```

Find employees who work on any project

If the answer to the subquery is not empty - then the EXISTS predicate returns TRUE
Tests for empty relations

Is this query correlated?

Exists: Example

- Retrieve the name of each employee who has a dependent with the same first name as the employee.

```
SELECT   FNAME, LNAME
FROM     EMPLOYEE AS E
WHERE    EXISTS  (SELECT      *
                  FROM        DEPENDENT
                  WHERE       E.SSN = ESSN AND
                      FNAME=DEPENDENT_NAME)
```

Correlated?

Not Exists: Example

- Retrieve the names of employees who have no dependents.

```
SELECT      FNAME, LNAME
FROM        EMPLOYEE
WHERE       NOT EXISTS   (SELECT        *
                          FROM  DEPENDENT
                          WHERE SSN = ESSN)
```

Exists and Unique

Four predicates can be placed in front of a subquery in SQL:

EXISTS (subquery)
NOT EXISTS (subquery)
UNIQUE (subquery) -- tests if there are any duplicate tuples
NOT UNIQUE (subquery) with the obvious semantics

Unique/Not Unique: Exercise

- Find employees who have more than one dependent.

```
SELECT      E.FName, E.LName
FROM        Employee AS E
WHERE       NOT UNIQUE
```

Warning: using UNIQUE and NOT UNIQUE in the WHERE clause is NOT supported in SQLite. The example above will give you a syntax error in SQLite

In-Class Exercise #2

- Find the names of employees who work for some of the projects controlled by the research department (use subquery)

ANY and ALL

- Use to compare a single value with a table of values
  - This table must have only one attribute and the domain must be the same as well
  - Typically associated with a subquery, in the WHERE clause

- Warning: using ANY or ALL in the WHERE clause is NOT supported in SQLite. The examples in the next couple slides will give you a syntax error in SQLite

All or Any: Example

```
SELECT    E.FName, E.LName
FROM      Employee AS E
WHERE     E.Salary <= ALL (SELECT Salary
                           FROM   Employee)
```

Find employees with the lowest salary

This predicate must be true for all salaries returned by the subquery

All or Any: General Form

```
SELECT    E.FName, E.LName
FROM      Employee AS E
WHERE     E.Salary  >= ALL  (SELECT   Salary
                             FROM     Employee);
```

This form of WHERE clause (with subquery) is:

attribute-name comparator ALL (subquery)
or
attribute-name comparator ANY (subquery)

All or Any: Example and Exercises

- A = 1, Subquery =

B
1
2
3

1. A <= All <Subquery>
2. A != Any <Subquery>
3. A != All <Subquery>
4. A >= Any <Subquery>
5. A < Any <Subquery>
6. (A + 1) > All<Subquery>

(A<=1) AND (A<=2) AND (A<=3)

(A!=1) OR (A!=2) OR (A!=3)

Summary of Today's Lecture

- Extension to Where
  - Boolean and comparator
  - Subquery
- SQL exercises