Lesson 7.2: Complex SQL

CSC430/530 – DATABASE MANAGEMENT SYSTEMS

OUTLINE

- Multisets & set operations.
- Pattern matching & additional operators.
- Three-valued logic.
- Nested queries.
- Exists, explicit sets & renaming.
- Joined tables.
- Aggregate functions.
- Group by & Having.

MULTISETS, SETS & SET OPERATIONS

- SQL treats tables as **multisets**.
  - **Duplicated** tuples *can* appear in the **result** of a query.
- **DISTINCT** is used in **SELECT** clause to **eliminate** duplicates in the query result
- For example:

```
SELECT Essn
FROM dependent
```

Essn
123456789
123456789
123456789
333445555
333445555
333445555
987654321

```
SELECT DISTINCT Essn
FROM dependent
```

Essn
123456789
333445555
987654321

DEPENDENT

Essn	Dependent_name	Sex	Bdate	Relationship
123456789	Alice	F	1988-12-30	Daughter
123456789	Elizabeth	F	1967-05-05	Spouse
123456789	Michael	M	1988-01-04	Son
333445555	Alice	F	1986-04-05	Daughter
333445555	Joy	F	1958-05-03	Spouse
333445555	Theodore	M	1983-10-25	Son
987654321	Abner	M	1942-02-28	Spouse

MULTISETS, SETS & SET OPERATIONS

- SQL supports **set operation** commands.
  - Duplicates are **eliminated** in the results of set operation queries.
  - **UNION, INTERSECT, EXCEPT** (*set difference*).

```
SELECT * FROM student
    UNION
SELECT * FROM instructor;
```

Fn	Ln
Susan	Yao
Ramesh	Shah
Johnny	Kohler
Barbara	Jones
Amy	Ford
Jimmy	Wang
Ernest	Gilbert
John	Smith
Ricardo	Browne
Francis	Johnson

```
SELECT * FROM student
    INTERSECT
SELECT * FROM instructor;
```

Fn	Ln
Susan	Yao
Ramesh	Shah

```
SELECT * FROM student
    EXCEPT
SELECT * FROM instructor;
```

Fn	Ln
Johnny	Kohler
Barbara	Jones
Amy	Ford
Jimmy	Wang
Ernest	Gilbert

STUDENT

Fn	Ln
Susan	Yao
Ramesh	Shah
Johnny	Kohler
Barbara	Jones
Amy	Ford
Jimmy	Wang
Ernest	Gilbert

INSTRUCTOR

Fname	Lname
John	Smith
Ricardo	Browne
Susan	Yao
Francis	Johnson
Ramesh	Shah

MULTISETS, SETS & SET OPERATIONS

- Adding **ALL** to **set operations** turn them into **multiset** operations.
  - Duplicates are **not eliminated**.
  - **UNION ALL, INTERSECT ALL, EXCEPT ALL.**

```
SELECT * FROM student
    UNION
SELECT * FROM instructor;
```

Fn	Ln
Susan	Yao
Ramesh	Shah
Johnny	Kohler
Barbara	Jones
Amy	Ford
Jimmy	Wang
Ernest	Gilbert
John	Smith
Ricardo	Browne
Francis	Johnson

```
SELECT * FROM student
    UNION ALL
SELECT * FROM instructor;
```

Fn	Ln
Susan	Yao
Ramesh	Shah
Johnny	Kohler
Barbara	Jones
Amy	Ford
Jimmy	Wang
Ernest	Gilbert
John	Smith
Ricardo	Browne
Francis	Johnson
Susan	Yao
Ramesh	Shah

STUDENT

Fn	Ln
Susan	Yao
Ramesh	Shah
Johnny	Kohler
Barbara	Jones
Amy	Ford
Jimmy	Wang
Ernest	Gilbert

INSTRUCTOR

Fname	Lname
John	Smith
Ricardo	Browne
Susan	Yao
Francis	Johnson
Ramesh	Shah

MULTISETS, SETS & SET OPERATIONS

Query 1:

Retrieve the salary of every employee;

Retrieve all distinct salary values.

Query 2:

Make a list of all project numbers for projects that involve an employee whose last name is ‘Smith’, either as a worker or as a manager of the department that controls the project.

MULTISETS, SETS & SET OPERATIONS

Query 1:

Retrieve the salary of every employee;

```
SELECT Salary FROM EMPLOYEE;
```

Retrieve all distinct salary values.

```
SELECT DISTINCT Salary FROM EMPLOYEE;
```

Query 2:

Make a list of all project numbers for projects that involve an employee whose last name is 'Smith', either as a worker or as a manager of the department that controls the project.

```
(SELECT W.Pno FROM WORKS_ON W, EMPLOYEE E
 WHERE W.Essn = E.Ssn AND E.Lname = 'Smith')
UNION
(SELECT P.Pnumber FROM PROJECT P, DEPARTMENT D, EMPLOYEE E
 WHERE P.Dnum = D.Dnumber AND D.Mgr_ssn = E.Ssn AND E.Lname = 'Smith');
```

PATTERN MATCHING & ADDITIONAL OPERATORS

- **LIKE** is used for substring **pattern matching**.
  - "_" replaces a **single** character, "%" replaces an **arbitrary number** of characters.

```
SELECT Pname, Pnumber
FROM project
WHERE Pname LIKE "Product_"
```

Pname	Pnumber
ProductX	1
ProductY	2
ProductZ	3

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

```
SELECT Fname, Ssn
FROM employee
WHERE Fname LIKE "J%"
```

Fname	Ssn
John	123456789
Jennifer	987654321
Joyce	453453453
James	888665555

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	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	1968-01-19	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

PATTERN MATCHING & ADDITIONAL OPERATORS

Query 3.1:

Retrieve all employees whose address is in Houston, Texas.

Query 3.2:

Find all employees who were born during the 1970s.

PATTERN MATCHING & ADDITIONAL OPERATORS

Query 3.1:

Retrieve all employees whose address is in Houston, Texas.

```
SELECT Fname, Lname
FROM EMPLOYEE
WHERE Address LIKE '%Houston, TX%';
```

Query 3.2:

Find all employees who were born during the 1970s.

```
SELECT Fname, Lname
FROM EMPLOYEE
WHERE Bdate LIKE '197_______';
/*                197x-xx-xx */
```

PATTERN MATCHING & ADDITIONAL OPERATORS

- **Additional** operators can be used in a query.
  - Arithmetic operators + – * /
  - **BETWEEN** operator (equivalent to <= AND >=).
- **ORDER BY** is used to order tuples in a query result.
  - **DESC** orders in descending order (from high to low)
  - **ASC** orders in ascending order (from low to high)

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
987654321	Abner	M	1942-02-28	Spouse
123456789	Michael	M	1988-01-04	Son
123456789	Alice	F	1988-12-30	Daughter
123456789	Elizabeth	F	1967-05-05	Spouse

Dependent_name	Bdate
Alice	1988-12-30
Alice	1986-04-05
Michael	1988-01-04
Theodore	1983-10-25

```
SELECT Dependent_name, Bdate
FROM dependent
WHERE bdate BETWEEN '1980-01-01' AND '1990-01-01'
ORDER BY Dependent_name ASC;
```

- same as -

```
SELECT Dependent_name, Bdate
FROM dependent
WHERE bdate >= '1980-01-01' AND bdate <= '1990-01-01'
ORDER BY Dependent_name ASC;
```

PATTERN MATCHING & ADDITIONAL OPERATORS

Dependent_name	Bdate
Alice	1988-12-30
Alice	1986-04-05
Michael	1988-01-04
Theodore	1983-10-25

```
SELECT Dependent_name, Bdate
FROM dependent
WHERE bdate BETWEEN '1980-01-01' AND '1990-01-01'
ORDER BY Dependent_name ASC;
```

Dependent_name	Bdate
Theodore	1983-10-25
Michael	1988-01-04
Alice	1986-04-05
Alice	1988-12-30

```
SELECT Dependent_name, Bdate
FROM dependent
WHERE bdate BETWEEN '1980-01-01' AND '1990-01-01'
ORDER BY Dependent_name DESC, Bdate ASC;
```

PATTERN MATCHING & ADDITIONAL OPERATORS

Query 4:
Show the resulting salaries if every employee working on the ‘Product’ project is given a 10% raise.

Query 5:
Retrieve all employees in department 5 whose salary is between $30,000 and $40,000.

Query 6:
Retrieve a list of employees and the projects they are working on, ordered by department and, within each department, ordered alphabetically by last name, then first name.

PATTERN MATCHING & ADDITIONAL OPERATORS

Query 4:

Show the resulting salaries if every employee working on the 'Product' project is given a 10% raise.

```
SELECT E.Fname, E.Lname, 1.1 * E.Salary AS Increased_sal
FROM EMPLOYEE AS E, WORKS_ON AS W, PROJECT AS P
WHERE E.Ssn=W.Essn AND W.Pno=P.Pnumber AND P.Pname='ProductX';
```

Query 5:

Retrieve all employees in department 5 whose salary is between $30,000 and $40,000.

```
SELECT * FROM EMPLOYEE
WHERE (Salary BETWEEN 30000 AND 40000) AND Dno = 5;
```

Query 6:

Retrieve a list of employees and the projects they are working on, ordered by department and, within each department, ordered alphabetically by last name, then first name.

```
SELECT D.Dname, E.Lname, E.Fname, P.Pname
FROM DEPARTMENT AS D, EMPLOYEE AS E, WORKS_ON AS W, PROJECT AS P
WHERE D.Dnumber=E.Dno AND E.Ssn=W.Essn AND W.Pno=P.Pnumber
ORDER BY D.Dname, E.Lname, E.Fname;
```

THREE-VALUED LOGIC

- **NULL** is used in SQL to represent **missing values**.
  - Unknown, unavailable, or not applicable.
- When comparing attributes with **NULL** value, the **result** is considered to be ***UNKNOWN***
  - Could be ***TRUE*** or ***FALSE***.
- SQL uses **three-valued logic**: **TRUE**, **FALSE**, **UNKNOWN**.
  - Only combinations of tuples that evaluate to **TRUE** in **WHERE** clause condition are selected (not FALSE and not UNKNOWN)

AND	TRUE	FALSE	UNKNOWN
TRUE	TRUE	FALSE	UNKNOWN
FALSE	FALSE	FALSE	FALSE
UNKNOWN	UNKNOWN	FALSE	UNKNOWN

OR	TRUE	FALSE	UNKNOWN
TRUE	TRUE	TRUE	TRUE
FALSE	TRUE	FALSE	UNKNOWN
UNKNOWN	TRUE	UNKNOWN	UNKNOWN

NOT	
TRUE	FALSE
FALSE	TRUE
UNKNOWN	UNKNOWN

THREE-VALUED LOGIC

- **IS** & **IS NOT** are used to check if an attribute value is (or is not) equal to **NULL**.

```
SELECT Fname, Lname
FROM employee
WHERE super_ssn IS NULL;
```

Fname	Lname
James	Borg

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	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	1968-01-19	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

THREE-VALUED LOGIC

Query 7:

Retrieve the names of all employees who have supervisors.

THREE-VALUED LOGIC

Query 7:

Retrieve the names of all employees who have supervisors.

```
SELECT Fname, Lname
FROM employee
WHERE super_ssn IS NOT NULL;
```

NESTED QUERIES

- Queries can be **nested inside FROM** or **WHERE** clauses of other **queries**.
  - **Inner** (*nested*) query, **outer** query.

```
SELECT emp_names.Fname
FROM (SELECT Fname, Lname FROM employee) AS emp_names;
```

Fname
Alice
Alice
Michael
Theodore

```
SELECT Dlocation
FROM dept_locations
WHERE Dnumber IN (
    SELECT Dnumber
    FROM department
    WHERE Dname = "Research"
);
```

Dlocation
Bellaire
Houston
Sugarland

DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date
Research	5	333445555	1988-05-22
Administration	4	987654321	1995-01-01
Headquarters	1	888665555	1981-06-19

DEPT_LOCATIONS

Dnumber	Dlocation
1	Houston
4	Stafford
5	Bellaire
5	Sugarland
5	Houston

NESTED QUERIES

- **Correlated nested queries** – condition in the **WHERE** clause of **inner query** references some attribute of a relation declared in the **outer query**.
  - **Inner** (*nested*) query is evaluated once for each tuple (or combination of tuples) in the **outer query**.
  - **Aliasing** is recommended to use in nested queries.

NESTED QUERIES

- **Comparison operators** can be used with **ANY** or **ALL** quantifiers.

```
SELECT Fname, Lname
FROM employee
WHERE ssn = ANY (
    SELECT Essn
    FROM dependent
    WHERE Bdate > '1960-01-01'
);
```

Fname	Lname
John	Smith
Franklin	Wong

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	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	1968-01-19	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

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
987654321	Abner	M	1942-02-28	Spouse
123456789	Michael	M	1988-01-04	Son
123456789	Alice	F	1988-12-30	Daughter
123456789	Elizabeth	F	1967-05-05	Spouse

NESTED QUERIES

- **Comparison operators** can be used with **ANY** or **ALL** quantifiers.

```
SELECT Fname, Lname
FROM employee
WHERE dno = 5 AND
      salary >= ALL (
          SELECT salary
          FROM employee
          WHERE dno = 5
      );
```

Fname	Lname
Franklin	Wong

```
SELECT Fname, Lname
FROM employee
WHERE dno = 4 AND
      salary <= ALL (
          SELECT salary
          FROM employee
          WHERE dno = 4
      );
```

Fname	Lname
Ahmad	Jabbar
Alicia	Zelaya

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	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	1968-01-19	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

NESTED QUERIES

Query 8:

Make a list of all project numbers for projects that involve an employee whose last name is ‘Smith’, either as a worker or as a manager of the department that controls the project.

NESTED QUERIES

Query 8:

Make a list of all project numbers for projects that involve an employee whose last name is ‘Smith’, either as a worker or as a manager of the department that controls the project.

```
SELECT Pnumber
FROM PROJECT
WHERE Pnumber IN (
    SELECT W.Pno
    FROM WORKS_ON W, EMPLOYEE E
    WHERE W.Essn = E.Ssn AND E.Lname = 'Smith'
)
OR Pnumber IN (
    SELECT P.Pnumber
    FROM PROJECT P, DEPARTMENT D, EMPLOYEE E
    WHERE P.Dnum = D.Dnumber AND D.Mgr_ssn = E.Ssn AND E.Lname = 'Smith'
);
```

NESTED QUERIES

Query 9:

Retrieve the name of each employee who has a dependent with the same first name and is the same sex as the employee.

NESTED QUERIES

Query 9:

Retrieve the name of each employee who has a dependent with the same first name and is the same sex as the employee.

```
SELECT E.Fname, E.Lname
FROM EMPLOYEE AS E
WHERE E.Ssn IN (
    SELECT D.Essn
    FROM DEPENDENT AS D
    WHERE E.Fname = D.Dependent_name AND E.Sex = D.Sex
);

/* In general, a query written with nested select-from-where blocks and using the = or
IN comparison operators can always be expressed as a single block query.*/
SELECT E.Fname, E.Lname
FROM EMPLOYEE AS E, DEPENDENT AS D
WHERE E.Ssn = D.Essn AND E.Sex = D.Sex AND
      E.Fname = D.Dependent_name;
```

NESTED QUERIES

Query 10:

List the names of employees whose salary is greater than the salary of all the employees in department 5.

NESTED QUERIES

Query 10:

List the names of employees whose salary is greater than the salary of all the employees in department 5.

```
SELECT Lname, Fname, Salary
FROM EMPLOYEE
WHERE Salary > ALL (
    SELECT Salary
    FROM EMPLOYEE
    WHERE Dno = 5
);
```

EXISTS

- **EXISTS** is a Boolean function that is used in **WHERE** clause.
  - Typically used in conjunction with a **correlated nested query**.
- **EXISTS** checks whether the **result** of a **nested query** is **empty** (contains no tuples) or **not**.
  - **TRUE** if the nested query result contains at least **one tuple**.
  - **FALSE** if the nested query result contains **no tuples**.

```
SELECT Fname, Lname
FROM employee AS e
WHERE EXISTS (
  SELECT *
  FROM department AS d
  WHERE mgr_start_date < '1990-01-01'
    AND d.mgr_ssn = e.super_ssn
);
```

Fname	Lname
Franklin	Wong
Jennifer	Wallace
John	Smith
Joyce	English
Ramesh	Narayan

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	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	1968-01-19	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

DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date
Research	5	333445555	1988-05-22
Administration	4	987654321	1995-01-01
Headquarters	1	888665555	1981-06-19

EXISTS

- We can achieve the same result in multiple ways

Fname	Lname
Franklin	Wong
Jennifer	Wallace
John	Smith
Joyce	English
Ramesh	Narayan

```sql
SELECT Fname, Lname
FROM employee AS e
WHERE EXISTS (
  SELECT *
  FROM department AS d
  WHERE mgr_start_date < '1990-01-01'
    AND d.mgr_ssn = e.super_ssn
);
```

```sql
SELECT Fname, Lname
FROM employee
WHERE super_ssn = ANY (
    SELECT mgr_ssn
    FROM department
    WHERE mgr_start_date < '1990-01-01'
);
```

EXISTS

- Can use **NOT EXISTS** to test the opposite (i.e. returns true if result is no tuples).

```
SELECT fname, lname
FROM employee AS e
WHERE NOT EXISTS (
  SELECT *
  FROM department AS d
  WHERE mgr_start_date < '1990-01-01'
    AND d.mgr_ssn = e.super_ssn
);
```

Fname	Lname
James	Borg
Ahmad	Jabbar
Alicia	Zelaya

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	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	1968-01-19	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

DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date
Research	5	333445555	1988-05-22
Administration	4	987654321	1995-01-01
Headquarters	1	888665555	1981-06-19

EXISTS

```sql
SELECT fname, lname
FROM employee AS e
WHERE NOT EXISTS (
  SELECT *
  FROM department AS d
  WHERE mgr_start_date < '1990-01-01'
    AND d.mgr_ssn = e.super_ssn
);
```

Fname	Lname
James	Borg
Ahmad	Jabbar
Alicia	Zelaya

```sql
SELECT fname, lname
FROM employee AS e
WHERE NOT EXISTS (
  SELECT *
  FROM department AS d
  WHERE mgr_start_date < '1990-01-01'
    AND d.mgr_ssn = e.super_ssn
)
AND e.super_ssn IS NOT NULL;
```

Fname	Lname
Ahmad	Jabbar
Alicia	Zelaya

EXISTS

Query 11:

Alternative to Query 9 (i.e. Retrieve the name of each employee who has a dependent with the same first name and is the same sex as the employee), but with EXISTS.

Query 12:

Retrieve the names of employees who have no dependents.

EXISTS

Query 11:

Alternative to Query 9 (i.e. Retrieve the name of each employee who has a dependent with the same first name and is the same sex as the employee), but with EXISTS.

```
SELECT E.Fname, E.Lname
FROM EMPLOYEE AS E
WHERE EXISTS (
    SELECT *
    FROM DEPENDENT AS D
    WHERE E.Ssn=D.Essn AND E.Sex=D.Sex AND E.Fname=D.Dependent_name
);
```

Query 12:

Retrieve the names of employees who have no dependents.

```
SELECT Fname, Lname
FROM EMPLOYEE E
WHERE NOT EXISTS (
    SELECT *
    FROM DEPENDENT D
    WHERE E.Ssn = D.Essn
);
```

EXISTS

Query 13:

List the names of managers who have at least one dependent.

EXISTS

Query 13:

List the names of managers who have at least one dependent.

```
SELECT Fname, Lname
FROM EMPLOYEE E
WHERE EXISTS (
    SELECT *
    FROM DEPARTMENT DEP
    WHERE E.Ssn = DEP.Mgr_ssn
)
AND EXISTS (
    SELECT *
    FROM DEPENDENT D
    WHERE E.Ssn = D.Essn
);
```

EXPLICIT SETS & RENAMING

- Use **IN** to specify **explicit sets** of values for the **WHERE** clause.
  - Enclose values in **parentheses**.
- **AS** command is used to **rename** (*alias*) **relations** or **attributes** in appropriate part of a query.

```
SELECT Pname AS Project_Name,
       Dnum AS Dept_Num
FROM project
WHERE Dnum IN (1, 4);
```

Project_Name	Dept_Num
Reorganization	1
Computerization	4
Newbenefits	4

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

EXPLICIT SETS & RENAMING

- You can **not** use an attribute alias from your **select** clause in your **where** clause.
  - This is not supported since (when the query is being compiled and executed) the select alias takes effect **after** the where clause has been applied.
- To accomplish this, use **HAVING** instead of **WHERE** (we will see this command again in a later slide)
  - Aliasing happens **before** HAVING is evaluated, so it knows about the aliases from the select clause

```
SELECT Pname AS Project_Name,
       Dnum AS Dept_Num
FROM project
WHERE Dept_Num IN (1, 4);
```

```
SELECT Pname AS Project_Name,
       Dnum AS Dept_Num
FROM project
HAVING Dept_Num IN (1, 4);
```

EXPLICIT SETS & RENAMING

Query 14:

Retrieve Social Security numbers of all employees who work on project numbers 1, 2, or 3.

Query 15:

Retrieve last name of each employee and his/her supervisor while renaming the resulting attribute names as Employee_name and Supervisor_name.

EXPLICIT SETS & RENAMING

Query 14:

Retrieve Social Security numbers of all employees who work on project numbers 1, 2, or 3.

```
SELECT DISTINCT Essn
FROM WORKS_ON
WHERE Pno IN (1, 2, 3);
```

Query 15:

Retrieve last name of each employee and his/her supervisor while renaming the resulting attribute names as Employee_name and Supervisor_name.

```
SELECT E.Lname AS Employee_name, S.Lname AS Supervisor_name
FROM EMPLOYEE AS E, EMPLOYEE AS S
WHERE E.Super_ssn = S.Ssn;
```

JOINED TABLES

- **Tables** can be **joined** explicitly in **FROM** clause.

- **JOIN … ON**.

```
SELECT p.pname, p.pnumber, p.plocation, d.dname
FROM ( project p JOIN department d ON p.dnum = d.dnumber );
```

- **NATURAL JOIN**.

```
SELECT p.pname, p.pnumber, p.plocation, d.dname
FROM ( project p NATURAL JOIN (
            SELECT dname, dnumber AS dnum
            FROM department
        ) AS d
);
```

JOINED TABLES

Query 16:

Retrieve the name and address of every employee who works for the ‘Research’ department.

Query 17:

Rename the department number attribute of DEPARTMENT relation and then use a NATURAL JOIN with EMPLOYEE relation.

JOINED TABLES

Query 16:

Retrieve the name and address of every employee who works for the ‘Research’ department.

```
SELECT E.Fname, E.Lname, E.Address
FROM (
    EMPLOYEE E JOIN DEPARTMENT D ON E.Dno = D.Dnumber
)
WHERE D.Dname = 'Research';
```

Query 17:

Rename the department number attribute of DEPARTMENT relation and then use a NATURAL JOIN with EMPLOYEE relation.

```
SELECT E.Fname, E.Lname, E.Address
FROM (
    EMPLOYEE E NATURAL JOIN (
        SELECT D.Dname, D.Dnumber AS Dno, D.Mgr_ssn, D.Mgr_start_date
        FROM DEPARTMENT D
    ) AS DEPT
)
WHERE DEPT.Dname = 'Research';
```

JOINED TABLES

```
SELECT E.Fname, E.Lname, E.Address
FROM ( employee E JOIN department D ON E.Dno = D.Dnumber )
WHERE D.Dname = 'Research';
```

```
SELECT E.Fname, E.Lname, E.Address
FROM (
    employee E NATURAL JOIN (
        SELECT D.Dname, D.Dnumber AS Dno, D.Mgr_ssn, D.Mgr_start_date
        FROM department D
    ) AS DEPT
)
WHERE DEPT.Dname = 'Research';
```

```
SELECT E.Fname, E.Lname, E.Address
FROM employee e, department d
WHERE e.dno = d.dnumber AND d.dname = 'Research';
```

JOINED TABLES

- Multiway JOIN … ON.

```
FROM (
    (employee e JOIN project p ON e.dno = p.dnum)
    JOIN works_on w ON p.pnumber = w.pno
)
```

JOINED TABLES

Query 18:

For every project located in ‘Stafford’, list the project number, the controlling department number, and the department manager’s last name, address, and birth date.

JOINED TABLES

Query 18:

For every project located in 'Stafford', list the project number, the controlling department number, and the department manager's last name, address, and birth date.

```
SELECT P.Pnumber, D.Dnumber, E.Lname, E.Address, E.Bdate
FROM (
     (PROJECT P JOIN DEPARTMENT D ON P.Dnum = D.Dnumber)
     JOIN EMPLOYEE E ON D.Mgr_ssn = E.Ssn
)
WHERE P.Plocation = 'Stafford';
```

AGGREGATE FUNCTIONS

- **Aggregate functions** are used to **summarize** information from **multiple** tuples into a **single-tuple** summary.
  - Normally used in **SELECT** or **HAVING** clause.
- **COUNT**, **SUM**, **MAX**, **MIN**, **AVG**.
  - **COUNT** function returns the number of tuples or values as specified in a query.
  - **SUM, MIN**, **MAX,** and **AVG** are used with attributes.
- **NULL** values are discarded when **aggregate functions** are applied to a particular attribute.
  - Exception is **COUNT(*)** since it counts tuples instead of values.

AGGREGATE FUNCTIONS

```
SELECT COUNT(essn),
       SUM(hours)
FROM works_on
WHERE pno = 1;
```

COUNT(essn)	SUM(hours)
2	52.5

```
SELECT dname, mgr_start_date
FROM department
WHERE mgr_start_date = (
    SELECT MIN(mgr_start_date)
    FROM department
);
```

dname	mgr_start_date
Headquarters	1981-06-19

```
SELECT dname, mgr_start_date
FROM department
WHERE mgr_start_date = (
    SELECT MAX(mgr_start_date)
    FROM department
);
```

dname	mgr_start_date
Administration	1995-01-01

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

DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date
Research	5	333445555	1988-05-22
Administration	4	987654321	1995-01-01
Headquarters	1	888665555	1981-06-19

AGGREGATE FUNCTIONS

Query 19.1:

Find the sum of the salaries of all employees, the maximum salary, the minimum salary, and the average salary without attribute renaming.

Query 19.2:

Find the sum of the salaries of all employees, the maximum salary, the minimum salary, and the average salary with attribute renaming.

AGGREGATE FUNCTIONS

Query 19.1:

Find the sum of the salaries of all employees, the maximum salary, the minimum salary, and the average salary without attribute renaming.

```
SELECT SUM(Salary),
       MAX(Salary),
       MIN(Salary),
       AVG(Salary)
FROM EMPLOYEE;
```

Query 19.2:

Find the sum of the salaries of all employees, the maximum salary, the minimum salary, and the average salary with attribute renaming.

```
SELECT SUM(Salary) AS Total_Salary,
       MAX(Salary) AS Highest_Salary,
       MIN(Salary) AS Lowest_Salary,
       AVG(Salary) AS Average_Salary
FROM EMPLOYEE;
```

AGGREGATE FUNCTIONS

Query 20:

Find the sum of the salaries of all employees of the ‘Research’ department, as well as the maximum salary, the minimum salary, and the average salary in this department.

Query 21:

Count the number of distinct salary values in the database.

AGGREGATE FUNCTIONS

Query 20:

Find the sum of the salaries of all employees of the 'Research' department, as well as the maximum salary, the minimum salary, and the average salary in this department.

```
SELECT SUM(E.Salary),
       MAX(E.Salary),
       MIN(E.Salary),
       AVG(E.Salary)
FROM (
    EMPLOYEE E JOIN DEPARTMENT D ON E.Dno = D.Dnumber
)
WHERE D.Dname = 'Research';
```

Query 21:

Count the number of distinct salary values in the database.

```
SELECT COUNT(DISTINCT Salary)
FROM EMPLOYEE;
```

AGGREGATE FUNCTIONS

Query 22:

Retrieve the names of all employees who have two or more dependents.

AGGREGATE FUNCTIONS

Query 22:

Retrieve the names of all employees who have two or more dependents.

```
SELECT E.Lname, E.Fname
FROM EMPLOYEE E
WHERE (
    SELECT COUNT(*)
    FROM DEPENDENT D
    WHERE E.Ssn = D.Essn
) >= 2;
```

GROUP BY & HAVING

- **GROUP BY** is used to **partition** relation into **non-overlapping subsets**.
  - Each **partition** (*group*) consists of the tuples that have **same value** for some **grouping attribute(s)**.
- **GROUP BY** specifies the **grouping attribute(s)**, which also appear(s) in **SELECT** clause.
- **GROUP BY** is applied **after joining** tables if used in conjunction with **join condition**.

```
SELECT Pno, SUM(hours)
FROM works_on
GROUP BY Pno;
```

Pno	SUM(hours)
1	52.5
2	37.5
3	50.0
10	55.0
20	25.0
30	55.0

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

GROUP BY & HAVING

```
SELECT dno,
       salary,
       COUNT(*) AS emp_count
FROM employee
GROUP BY dno, salary;
```

dno	salary	emp_count
1	55000.00	1
4	25000.00	2
4	43000.00	1
5	25000.00	1
5	30000.00	1
5	38000.00	1
5	40000.00	1

```
SELECT dno,
       salary,
       COUNT(*) AS emp_count
FROM employee
GROUP BY dno;
```

dno	salary	emp_count
1	55000.00	1
4	43000.00	3
5	30000.00	4

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	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	1968-01-19	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

GROUP BY & HAVING

Query 23:

For each department, retrieve the department number, the number of employees in the department, and their average salary.

Query 24:

For each project, retrieve the project number, the project name, and the number of employees who work on that project.

GROUP BY & HAVING

Query 23:

For each department, retrieve the department number, the number of employees in the department, and their average salary.

```
SELECT Dno, COUNT(*), AVG(Salary)
FROM EMPLOYEE
GROUP BY Dno;
```

Query 24:

For each project, retrieve the project number, the project name, and the number of employees who work on that project.

```
SELECT P.Pnumber, P.Pname, COUNT(*)
FROM PROJECT P, WORKS_ON W
WHERE P.Pnumber = W.Pno
GROUP BY P.Pnumber, P.Pname;
```

GROUP BY & HAVING

- **HAVING** is used with **GROUP BY** to retrieve only those groups that **satisfy** certain **condition**.
- **HAVING** is applied **after** the **condition** is **evaluated** in **WHERE** clause.
  - **WHERE** clause is executed **first**, to select **individual tuples** or **joined tuples**;
  - **HAVING** clause is applied **later**, to select **individual groups** of **tuples**.

```
SELECT dno, salary, COUNT(*) AS emp_count
FROM employee
GROUP BY dno, salary;
```

dno	salary	emp_count
1	55000.00	1
4	25000.00	2
4	43000.00	1
5	25000.00	1
5	30000.00	1
5	38000.00	1
5	40000.00	1

```
SELECT dno, salary, COUNT(*) AS emp_count
FROM employee
GROUP BY dno, salary
HAVING salary > 30000;
```

dno	salary	emp_count
1	55000.00	1
4	43000.00	1
5	38000.00	1
5	40000.00	1

GROUP BY & HAVING

Query 25:

For each project on which more than two employees work, retrieve the project number, the project name, and the number of employees who work on the project.

GROUP BY & HAVING

Query 25:

For each project on which more than two employees work, retrieve the project number, the project name, and the number of employees who work on the project.

```
SELECT P.Pnumber,
       P.Pname,
       COUNT(*) AS emp_count
FROM project AS P,
     works_on AS W
WHERE P.Pnumber = W.Pno
GROUP BY P.Pnumber, P.Pname
HAVING emp_count > 2;
```

SUMMARY OF SQL QUERIES

- **General structure** of **retrieval** SQL query.
  - **SELECT** *<attribute and function list>*
    FROM *<table list>*
    [**WHERE** *<condition>*]
    [**GROUP BY** *<grouping attribute(s)>*]
    [**HAVING** *<group condition>*]
    [**ORDER BY** *<attribute list>*];
    - **SELECT** lists the attributes or functions to be retrieved.
    - **FROM** specifies all relations (tables) needed in the query, including joined relations.
    - **WHERE** specifies the conditions for selecting the tuples from these relations, including join conditions.
    - **GROUP BY** specifies grouping attributes.
    - **HAVING** specifies a condition on the groups being selected rather than on the individual tuples.
    - Aggregate functions (**COUNT**, **SUM**, **MIN**, **MAX**, **AVG**) are used in conjunction with grouping, or applied to all the selected tuples in a query without a GROUP BY clause.
    - **ORDER BY** specifies an order for displaying the result of a query.