SQL: Schema Definition, Basic Constraints, and Queries

SQL INTRODUCTION

- SQL stands for **Structured Query Language**
- Standard language for querying and manipulating data

Data Definition Language (DDL)

- Create, alter, delete tables and their attributes

Data Manipulation Language (DML)

- Query one or more tables
- Insert, delete, modify tuples in tables

Many standards out there:

ANSI SQL, SQL92 (a.k.a. SQL2), SQL99 (a.k.a. SQL3), . . .

SQL CONSTRAINTS

- Assigning Names to Constraints

```
CONSTRAINT deptPK PRIMARY KEY(Dnumber)
CONSTRAINT deptSK UNIQUE(Dname)
```

ALTER COMMAND

- The definition of table can be changed using ALTER command
- ALTER can be used to add an attribute to the relation
  - Initially, the new attribute will have NULLs in all the tuples of the relation
  - NOT NULL constraint is *not allowed* for such an attribute

Example :

ALTER TABLE EMPLOYEE ADD JOB VARCHAR(12);

The database user have to enter a value for the new attribute JOB for each EMPLOYEE tuple.

DROP COMMAND

- Drop Command is used to delete schema or named schema elements such as table, domains, or constraints

Example:

```
DROP TABLE  DEPENDENT;
DROP TABLE EMPLOYEE CASCADE;
DROP SCHEMA COMPANY;
```

In SQL-Server (T-SQL), DROP TABLE cannot be used to drop a table that is referenced by a FOREIGN KEY. The referencing FOREIGN KEY or the referencing table must first be dropped.

SQL QUERIES

Basic form

```
SELECT  <attributes>
FROM    <one or more relations>
WHERE   <conditions>
```

Not same *as* the Select σ operation of the relational algebra

Try SQLFiddle for online SQL practice
http://sqlfiddle.com/

https://www.db-fiddle.com/

RELATIONAL DATABASE SCHEMA

EMPLOYEE

FNAME	MINIT	LNAME	SSN	BDATE	ADDRESS	SEX	SALARY	SUPERSSN	DNO

DEPARTMENT

DNAME	DNUMBER	MGRSSN	MGRSTARTDATE

DEPT_LOCATIONS

DNUMBER	DLOCATION

PROJECT

PNAME	PNUMBER	PLOCATION	DNUM

WORKS_ON

ESSN	PNO	HOURS

DEPENDENT

ESSN	DEPENDENT_NAME	SEX	BDATE	RELATIONSHIP

SIMPLE SQL QUERIES

Retrieve the details of employees who work under the supervision of employee with id=333445555

```
SELECT   *
FROM     EMPLOYEE
WHERE    Super_ssn = 333445555
```

"selection"

$\sigma_{\text{Super\_SSN} = 333445555}$ (EMPLOYEE)

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

SIMPLE SQL QUERIES

Retrieve the birthdate and address of the employee whose name is 'John Smith'.

```
SELECT    BDATE, ADDRESS
FROM      EMPLOYEE
WHERE     FNAME='John' AND LNAME='Smith'
```

"selection" and "projection"

$$\pi_{\text{BDATE, ADDRESS}}(\sigma_{\text{FNAME='John' AND LNAME='Smith'}}(\text{EMPLOYEE}))$$

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

SQL QUERIES

- In SQL, the result can have **duplicate tuples**
  - SQL relation is a ***multi-set*** **(bag)** of tuples; *not* a set of tuples

```
SELECT   Salary
FROM     Employee
```

Salary
30000
40000
25000
43000
38000
25000
25000
55000

ELIMINATING DUPLICATES

```
SELECT   Salary
FROM     Employee
```

Salary
30000
40000
25000
43000
38000
25000
25000
55000

```
SELECT DISTINCT Salary
FROM     Employee
```

Salary
30000
40000
25000
43000
38000
55000

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

JOIN OPERATION

Retrieve the name of all projects

SELECT PNAME
FROM PROJECT **JOIN** DEPARTMENT **ON** DNUM=DNUMBER

$$DEPT\_MGR \leftarrow \pi_{PNAME} (PROJECT \bowtie_{DNUM=DNUMBER} DEPARTMENT)$$

DEPARTMENT

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

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

JOIN OPERATION

Retrieve the name of all projects **that are offered by 'Research' department.**

SELECT PNAME
FROM PROJECT **JOIN** DEPARTMENT **ON** DNUM=DNUMBER
WHERE **DNAME='Research'**

$D\_MGR \leftarrow \pi_{PNAME} (PROJECT \bowtie_{DNUM=DNUMBER} (\sigma_{DNAME='Research'}(DEPARTMENT))$

DEPARTMENT

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

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

JOIN OPERATION

Retrieve the name of all projects that are offered by 'Research' department.

SELECT PNAME
FROM **PROJECT , DEPARTMENT**
WHERE DNAME='Research' **and** **DNUM=DNUMBER**

join condition

DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date
Research	5	333445555	1988-0
Administration	4	987654321	1995-0
Headquarters	1	888665555	1981-0

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

DIFFERENT JOINS IN SQL

```
SELECT  PNAME
FROM    PROJECT JOIN DEPARTMENT ON DNUM=DNUMBER
WHERE   DNAME='Research'
```

```
SELECT  PNAME
FROM    PROJECT , DEPARTMENT
WHERE   DNAME='Research' and DNUM=DNUMBER
```

No Natural Join in T-SQL

UNSPECIFIED WHERE-clause

Retrieve the SSN values for all employees.

- **SELECT SSN**
- **FROM EMPLOYEE**

Missing WHERE-clause

- indicates there is no condition and is same as WHERE TRUE

If there is no join condition, then we get *CARTESIAN PRODUCT*

- **SELECT SSN, DNAME**
- **FROM EMPLOYEE, DEPARTMENT**

Ssn	Dname
123456789	Research
333445555	Research
999887777	Research
987654321	Research
666884444	Research
453453453	Research
987987987	Research
888665555	Research
123456789	Administration
333445555	Administration
999887777	Administration
987654321	Administration
666884444	Administration
453453453	Administration
987987987	Administration
888665555	Administration
123456789	Headquarters
333445555	Headquarters
999887777	Headquarters
987654321	Headquarters
666884444	Headquarters
453453453	Headquarters
987987987	Headquarters
888665555	Headquarters

Example Queries

List the employees name and the department name that they manage.

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

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								4321	4
Jennifer								5555	4
Ramesh								5555	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

$$\text{Temp} \leftarrow (\text{Employee} \bowtie_{\text{Ssn=Mgr\_Ssn}} \text{Department})$$

$$\text{Result} \leftarrow \pi_{\text{Fname, Minit, Lname, Dname}}(\text{Temp})$$

Example Queries

List the employees name and the department name that they manage.

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

```
SELECT Fname, Lname, Dname
FROM Employee Join Department ON Ssn=Mgr_ssn
```

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

JOIN OPERATION

For projects located in 'Stafford', list the project no, the controlling department no, and the department manager's last name, address and birthdate.

$SP \leftarrow \sigma_{PLOCATION='Stanfford'}, PROJECT)$

$R1 \leftarrow EMPLOYEE \bowtie_{SSN=Mgr\_ssn} (SP \bowtie_{Dnum=Dnumber} Department)$

$Result \leftarrow \pi_{FPnumber, Dnumber, Lname, Address, Bdate} (R1)$

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, Ho				
Alicia	J	Zelaya	999887777	1968-01-19	3321 Castle,				
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Be				

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

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

JOIN OPERATION

For projects located in 'Stafford', list the project no, the controlling department no, and the department manager's last name, address and birthdate.

```
SELECT    PNUMBER, DNUM, LNAME, ADDRESS, BDATE
FROM      ((PROJECT JOIN DEPARTMENT ON DNUM=DNUMBER)
                JOIN EMPLOYEE ON MGRSSN=SSN)
WHERE     PLOCATION = 'Stafford'
```

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, Ho				
Alicia	J	Zelaya	999887777	1968-01-19	3321 Castle,				
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Be				

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

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

JOIN OPERATION

For projects located in 'Stafford', list the project no, the controlling department no, and the department manager's last name, address and birthdate.

```
SELECT    PNUMBER, DNUM, LNAME, ADDRESS, BDATE
FROM      ((PROJECT JOIN DEPARTMENT ON DNUM=DNUMBER)
                JOIN EMPLOYEE ON MGRSSN=SSN )
WHERE     PLOCATION='Stafford'
```

Pnumber	Dnum	Lname	Address	Bdate
10	4	Wallace	291Berry, Bellaire, TX	1941-06-20
30	4	Wallace	291Berry, Bellaire, TX	1941-06-20

EMPLOY

Fname
John
Franklin
Alicia
Jennifer

DEPART

Dn			
Research	5	333445555	1988-05-22
Administration	4	987654321	1995-01-01
Headquarters	1	888665555	1981-06-19

			Dnum
			5
			5
			5
Computerization	10	Stafford	4
Reorganization	20	Houston	1
Newbenefits	30	Stafford	4

SET OPERATIONS

SQL set operations are

- Union operation (**UNION**),
- Set difference (**EXCEPT**)
- Intersection operation (**INTERSECT**)

Duplicate tuples are eliminated from the result

Requires *union compatible relations*

SET OPERATIONS

Make a list of Pname's for projects that involve an employee whose last name is 'Smith'

as a worker **OR**

as a manager of the department that controls the project.

EMPLOYEE

FNAME	MINIT	LNAME	SSN	BDATE	ADDRESS	SEX	SALARY	SUPERSSN	DNO

DEPARTMENT

DNAME	DNUMBER	MGRSSN	MGRSTARTDATE

PROJECT

PNAME	PNUMBER	PLOCATION	DNUM

DEPT_LOCATIONS

DNUMBER	DLOCATION

WORKS_ON

ESSN	PNO	HOURS

SET OPERATIONS

Make a list of Pname's for projects that involve an employee whose last name is 'Smith'

as a worker **OR**

as a manager of the department that controls the project.

```
(SELECT  PNAME
FROM     (PROJECT JOIN WORKS_ON ON PNUMBER=PNO ) JOIN
                    EMPLOYEE ON ESSN=SSN
WHERE    LNAME='Smith')
UNION
(SELECT  PNAME
         FROM     (PROJECT JOIN DEPARTMENT ON DNUM=DNUMBER ) JOIN
                            EMPLOYEE ON MGRSSN=SSN
         WHERE    LNAME='Smith')
```

EXAMPLE: Retrieve the SSN of employees who have no dependents.

$$ALL\_EMPS \leftarrow \pi_{SSN}(EMPLOYEE)$$

$$EMPS\_WITH\_DEPS(SSN) \leftarrow \pi_{ESSN}(DEPENDENT)$$

$$EMPS\_WITHOUT\_DEPS \leftarrow (ALL\_EMPS - EMPS\_WITH\_DEPS)$$

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

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

EXAMPLE: Retrieve the SSN of employees who have no dependents.

```
SELECT SSN FROM EMPLOYEE
EXCEPT
SELECT ESSN FROM DEPENDENT
```

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

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

EXAMPLE: Retrieve the names of employees who have no dependents.

ALL_EMPS ← π SSN(EMPLOYEE)

EMPS_WITH_DEPS(SSN) ← π ESSN(DEPENDENT)

EMPS_WITHOUT_DEPS ← (ALL_EMPS - EMPS_WITH_DEPS)

RESULT ← π LNAME, FNAME (EMPS_WITHOUT_DEPS * EMPLOYEE)

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

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

EXAMPLE: Retrieve the SSN of employees who has a daughter as well as a son

```
SELECT ESSN
FROM DEPENDENT
WHERE Relationship = 'Daughter' and Relationship = 'SON'
```

WHATS WRONG ??

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

EXAMPLE: Retrieve the SSN of employees who has a daughter as well as a son

```
SELECT ESSN
FROM DEPENDENT
WHERE Relationship = 'Daughter'

INTERSECT

SELECT ESSN
FROM DEPENDENT
WHERE Relationship = 'SON'
```

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

SET OPERATIONS

Find SSN of employees who work on project named ProductX **and** as well as on project named ProductY

EMPLOYEE

FNAME	MINIT	LNAME	SSN	BDATE	ADDRESS	SEX	SALARY	SUPERSSN	DNO

DEPARTMENT

DNAME	DNUMBER	MGRSSN	MGRSTARTDATE

PROJECT

PNAME	PNUMBER	PLOCATION	DNUM

DEPT_LOCATIONS

DNUMBER	DLOCATION

WORKS_ON

ESSN	PNO	HOURS

Lecture 7

ALIASES

- **In SQL**, we can use the **same name for two or more attributes** as long as the attributes are in ***different relations***

A query that refers to two attributes with the same name must *prefix* the relation's name to the attribute name

Example:

EMPLOYEE.DNO, DEPARTMENT.DNO

ALIASES

For each employee, retrieve the employee's name, and the name of his or her immediate supervisor.

Using **AS** keyword to specify aliases

```
SELECT   E.FNAME, E.LNAME, S.FNAME, S.LNAME
FROM     EMPLOYEE AS E, EMPLOYEE AS S
WHERE    E.SUPERSSN=S.SSN
```

```
SELECT   E.FNAME, E.LNAME, S.FNAME, S.LNAME
FROM     EMPLOYEE E S
WHERE    E.SUPERSSN=S.SSN
```

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address
John	B	Smith	123456789	1965-01-09	731 Fondren, Ho
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houst
Alicia	J	Zelaya	999887777	1968-01-19	3321 Castle, Sp
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellai
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Hu
Joyce	A	English	453453453	1972-07-31	5631 Rice, Hous
Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Hou
James	E	Borg	888665555	1937-11-10	450 Stone, Hou

E.Fname	E.Lname	S.Fname	S.Lname
John	Smith	Franklin	Wong
Franklin	Wong	James	Borg
Alicia	Zelaya	Jennifer	Wallace
Jennifer	Wallace	James	Borg
Ramesh	Narayan	Franklin	Wong
Joyce	English	Franklin	Wong
Ahmad	Jabbar	Jennifer	Wallace

ARITHMETIC OPERATIONS

Arithmetic operators '+', '-'. '*', and '/') can be applied to numeric values in an SQL query result

Give all employees who work on the 'ProductX' project a 10% raise.

```
SELECT    FNAME, LNAME, 1.1*SALARY
FROM      (WORKS_ON JOIN PROJECT ON PNO=PNUMBER)
          JOIN EMPLOYEE ON ESSN=SSN
WHERE     PNAME='ProductX'
```

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				M	40000	888665555	5
Alicia	J	Zelaya				F	25000	987654321	4
Jennifer	S	Wallace				F	43000	888665555	4
Ramesh	K	Narayan				M	38000	333445555	5
Joyce	A	English				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

PROJECT

PNAME	PNUMBER	PLOCATION	DNUM

WORKS_ON

ESSN	PNO	HOURS

ORDER BY

The **ORDER BY** clause **sort** the ***tuples*** in a query result

Retrieve a list of employees and the projects each works in, ordered by the employee's department number

```
SELECT      DNO, LNAME, FNAME, PNO
FROM        EMPLOYEE JOIN WORKS_ON ON SSN=ESSN
ORDER BY    DNO
```

EMPLOYEE

FNAME	MINIT	LNAME	SSN	BDATE	ADDRESS	SEX	SALARY	SUPERSSN	DNO

DEPARTMENT

DNAME	DNUMBER	MGRSSN	MGRSTARTDATE

PROJECT

PNAME	PNUMBER	PLOCATION	DNUM

DEPT_LOCATIONS

DNUMBER	DLOCATION

WORKS_ON

ESSN	PNO	HOURS

ORDER BY

The **ORDER BY** clause **sort** the ***tuples*** in a query result

Retrieve a list of employees and the projects each works in, ordered by the employee's department number

```
SELECT      DNO, LNAME, FNAME, PNO
FROM        EMPLOYEE JOIN WORKS_ON ON SSN=ESSN
ORDER BY    DNO
```

The default order is in ***ascending order*** of values

We can specify the keyword **DESC** if we want a descending order

- **ORDER BY** Dname **DESC,** Lname **ASC**

ORDER BY

Retrieve a list of Male employees and the projects each works in, ordered by the employee's department **name**, **and within each department ordered alphabetically by employee last name, then first name.**

```
SELECT      DNAME, LNAME, FNAME, PNAME
FROM        ((DEPARTMENT JOIN EMPLOYEE ON DNUMBER=DNO )
                JOIN WORKS_ON ON SSN=ESSN )
                JOIN PROJECT ON PNO=PNUMBER)
WHERE        SEX = 'MALE'
ORDER BY    DNAME, LNAME, FNAME
```

SQL QUERIES

- Retrieve the names of all employees who do not have supervisors.
- **SELECT FNAME, LNAME**
  FROM EMPLOYEE
  WHERE SUPERSSN IS NULL
- **<u>Note:</u> If a join condition is specified, tuples with NULL values for the join attributes are not included in the result**

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

SUBSTRING COMPARISON

LIKE operator is used to compare partial strings

Two reserved characters are used:

- '%' (or '*' in some implementations) replaces an arbitrary number of characters, and
- '_' replaces a single arbitrary character

Retrieve all employees whose address is in Houston, Texas.

-
```
SELECT    FNAME, LNAME
FROM      EMPLOYEE
WHERE     ADDRESS LIKE '%Houston,TX%'
```

SUBSTRING COMPARISON

Retrieve all employees who were born during the 1950s.

- **SELECT** **FNAME, LNAME**
  FROM **EMPLOYEE**
  WHERE **BDATE LIKE** ‘195_',.

LIKE operator allows us to get around the fact that each value is considered atomic and indivisible

Hence, in SQL, character string attribute values are not atomic

AGGREGATE FUNCTIONS

- Include **COUNT**, **SUM**, **MAX**, **MIN**, and **AVG**
- Find the maximum salary, the minimum salary, and the average salary among all employees.

```
SELECT   MAX(SALARY), MIN(SALARY), AVG(SALARY)
FROM     EMPLOYEE
```

$\mathcal{F}_{\text{SUM Salary, AVERAGE Salary, MIN Salary}}$ (EMPLOYEE)

Some SQL implementations *may not allow more than one function* in the SELECT-clause

AGGREGATE FUNCTIONS

Retrieve the no. of employees in the ‘Administration' department

```
SELECT   COUNT (*)
FROM     EMPLOYEE JOIN DEPARTMENT ON DNO=DNUMBER
WHERE    DNAME=‘Administration’
```

EMPLOYEE

FNAME	MINIT	LNAME	SSN	BDATE	ADDRESS	SEX	SALARY	SUPERSSN	DNO

DEPARTMENT

DNAME	DNUMBER	MGRSSN	MGRSTARTDATE

PROJECT

PNAME	PNUMBER	PLOCATION	DNUM

DEPT_LOCATIONS

DNUMBER	DLOCATION

WORKS_ON

ESSN	PNO	HOURS

Aggregate Functions $\mathcal{F}$

- $\mathcal{F}_{\text{MAX Salary}}$ (EMPLOYEE)
- $\mathcal{F}_{\text{MIN Salary}}$ (EMPLOYEE)
- $\mathcal{F}_{\text{SUM Salary, AVERAGE Salary}}$ (EMPLOYEE)
- $\mathcal{F}_{\text{COUNT SSN}}$ (EMPLOYEE)

COUNT (*) returns the no. of rows in the result of the query ***(it counts without removing duplicates)***

NULL values are **discarded** when aggregate functions are applied to a particular column (attribute).

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

AGGREGATE EXAMPLE

Count the number of distinct salary values in the database

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

NULL values are **discarded** when aggregate functions are applied to a particular attribute.

Aggregate functions are allowed only in the SELECT and the HAVING clause of a SQL statement.

GROUPING

GROUP BY-clause specifies the grouping attributes

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

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

GROUPING

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

SELECT	**DNO, COUNT (*), AVG (SALARY)**
FROM	**EMPLOYEE**
GROUP BY	**DNO**

$$\text{DNO}\ \mathcal{F}_{\text{COUNT SSN, AVERAGE Salary}}\ \text{EMPLOYEE}$$

Fname	Minit	Lname	Ssn	...	Salary	Super_ssn	Dno
John	B	Smith	123456789		30000	333445555	5
Franklin	T	Wong	333445555		40000	888665555	5
Ramesh	K	Narayan	666884444		38000	333445555	5
Joyce	A	English	453453453	...	25000	333445555	5
Alicia	J	Zelaya	999887777		25000	987654321	4
Jennifer	S	Wallace	987654321		43000	888665555	4
Ahmad	V	Jabbar	987987987		25000	987654321	4
James	E	Borg	888665555		55000	NULL	1

Grouping EMPLOYEE tuples by the value of Dno

Dno	Count_ssn	Average_salary
5	4	33250
4	3	31000
1	1	55000

Grouping with Aggregation

$_{DNO}\mathcal{F}_{COUNT\ SSN,\ AVERAGE\ Salary}$ EMPLOYEE

Dno	Count_ssn	Average_salary
5	4	33250
4	3	31000
1	1	55000

$\mathcal{F}_{COUNT\ SSN,\ AVERAGE\ Salary}$ EMPLOYEE

Count_ssn	Average_salary
8	35125

$\rho_{R(Dno,\ No\_of\_employees,\ Average\_sal)}$ ($_{DNO}\mathcal{F}_{COUNT\ SSN,\ AVERAGE\ Salary}$ EMPLOYEE)

R

Dno	No_of_employees	Average_sal
5	4	33250
4	3	31000
1	1	55000

GROUPING

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

EMPLOYEE

FNAME	MINIT	LNAME	SSN	BDATE	ADDRESS	SEX	SALARY	SUPERSSN	DNO

DEPARTMENT

DNAME	DNUMBER	MGRSSN	MGRSTARTDATE

PROJECT

PNAME	PNUMBER	PLOCATION	DNUM

DEPT_LOCATIONS

DNUMBER	DLOCATION

WORKS_ON

ESSN	PNO	HOURS

GROUPING

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

```
SELECT      PNUMBER, PNAME, COUNT (*)
FROM        PROJECT, WORKS_ON
WHERE       PNUMBER=PNO
GROUP BY    PNUMBER, PNAME
```

	PNUMBER	PNAME	count
1	1	ProductX	2
2	2	ProductY	3
3	3	ProductZ	2
4	10	Compu...	3
5	20	Reorga...	3
6	30	Newbe...	3

Grouping & Aggregate functions are applied *after* the joining two relations

Select clause can only include the grouping attributes and aggregate functions

Example: Retrieve the names of all employees with two or more dependents

$$T1(Ssn, No\_of\_dependents) \leftarrow Essn\ \mathcal{F}_{COUNT\ Dependent\_name}\ DEPENDENT$$

$$T2 \leftarrow \sigma_{No\_of\_dependents > 1}(T1)$$

$$RESULT \leftarrow \pi_{LNAME,\ FNAME}\ (T2 * EMPLOYEE)$$

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

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

HAVING-CLAUSE

HAVING-clause specify a selection condition on groups

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

PROJECT

PNAME	PNUMBER	PLOCATION	DNUM

WORKS_ON

ESSN	PNO	HOURS

```
SELECT      PNUMBER, PNAME, COUNT (*)
FROM        PROJECT JOIN WORKS_ON ON
                 PNUMBER=PNO
GROUP BY    PNUMBER, PNAME
HAVING      COUNT (*) > 2
```

HAVING-CLAUSE

Pname	Pnumber	...	Essn	Pno	Hours
ProductX	1		123456789	1	32.5
ProductX	1		453453453	1	20.0
ProductY	2		123456789	2	7.5
ProductY	2		453453453	2	20.0
ProductY	2		333445555	2	10.0
ProductZ	3		666884444	3	40.0
ProductZ	3		333445555	3	10.0
Computerization	10	...	333445555	10	10.0
Computerization	10		999887777	10	10.0
Computerization	10		987987987	10	35.0
Reorganization	20		333445555	20	10.0
Reorganization	20		987654321	20	15.0
Reorganization	20		888665555	20	NULL
Newbenefits	30		987987987	30	5.0
Newbenefits	30		987654321	30	20.0
Newbenefits	30		999887777	30	30.0

After applying the WHERE clause but before applying HAVING

These groups are not selected by the HAVING condition of Q26.

PNUMBER	PNAME	count
2	ProductY	3
10	Computerization	3
20	Reorganization	3
30	Newbenefits	3

HAVING-CLAUSE

Pname	Pnumber	...	Essn	Pno	Hours
ProductY	2		123456789	2	7.5
ProductY	2		453453453	2	20.0
ProductY	2		333445555	2	10.0
Computerization	10		333445555	10	10.0
Computerization	10	...	999887777	10	10.0
Computerization	10		987987987	10	35.0
Reorganization	20		333445555	20	10.0
Reorganization	20		987654321	20	15.0
Reorganization	20		888665555	20	NULL
Newbenefits	30		987987987	30	5.0
Newbenefits	30		987654321	30	20.0
Newbenefits	30		999887777	30	30.0

After applying the HAVING clause condition

Pname	Count (*)
ProductY	3
Computerization	3
Reorganization	3
Newbenefits	3

Result of Q26
(Pnumber not shown)

General form of Grouping and Aggregation

```
SELECT    S
FROM      R1,…,Rn
WHERE     C1
GROUP BY  a1,…,ak
HAVING    C2
```

Evaluation steps:

- Evaluate FROM-WHERE, apply condition C1
- Group by the attributes $a_1,\ldots,a_k$
- Apply condition C2 to each group (may have aggregates)
- Compute aggregates in S and return the result

Example

- List the employees name and the department name that they manage.
- Temp ← (Employee $\bowtie_{Ssn=Mgr\_Ssn}$ Department)
- Result ← $\pi_{Fname, Minit, Lname, Dname}$(Temp)

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

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

Left Outer Join

- List the employees name and the department name that they manage. **If they don't manage one, then indicate this with a null value.**
- Temp ← (Employee $\overset{\text{⟕}}{\underset{Ssn=Mgr\_Ssn}{}}$ Department)
- Result ← $\pi_{Fname,\ Minit,\ Lname,\ Dname}$(Temp)

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

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

Left Outer Join

- List the employees name and the department name that they manage. **If they don't manage one, then indicate this with a null value.**
- Temp ← (Employee $\overset{\text{⟕}}{\bowtie}_{Ssn=Mgr\_Ssn}$ Department)
- Result ← $\pi_{Fname,\ Minit,\ Lname,\ Dname}$(Temp)

RESULT

Fname	Minit	Lname	Dname
John	B	Smith	NULL
Franklin	T	Wong	Research
Alicia	J	Zelaya	NULL
Jennifer	S	Wallace	Administration
Ramesh	K	Narayan	NULL
Joyce	A	English	NULL
Ahmad	V	Jabbar	NULL
James	E	Borg	Headquarters

Right Outer Join

- List the employees name and the department name that they manage. If they don't manage one, then indicate this with a null value.
- Temp ← (**Department** $\bowtie_{Mgr\_Ssn = Ssn}$ **Employee**)
- Result ← $\pi_{Fname, Minit, Lname, Dname}$(Temp)

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

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

Full Outer Join

List the employees name and the department name that they manage. If they don't manage one **<u>or the department have no manager,</u>** then indicate this with a null value.

$$Temp \leftarrow Employee \ ⟗_{Ssn = Mgr\_Ssn} \ Department$$

$$Result \leftarrow \pi_{Fname,\ Lname,\ Dname}(Temp)$$

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
CS	6		

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

Full Outer Join vs Cartesian Product

What is the difference ?
OR …
are they same … ?

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
CS	6		

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

Outer Join Operation

- In INNER JOIN, tuples without a *matching* are eliminated from the join result
  - Tuples with null are also eliminated
  - This amounts to loss of information.

OUTER joins operations are used when we want to keep

- **all the tuples in R** in the join result , or
- **all the tuples in S** in the join result, or
- **all tuples in both relations R and S** in the join result

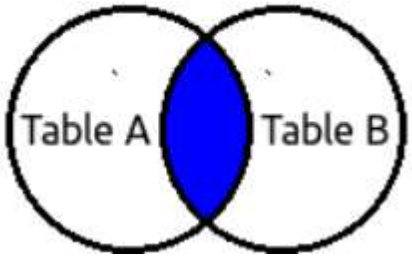
Outer Join Operation

- **Left outer join**: keeps every tuple in R, denoted as R ⟕ S
  - if no matching tuple is found in S, then the attributes of S in the join result are filled with null values.

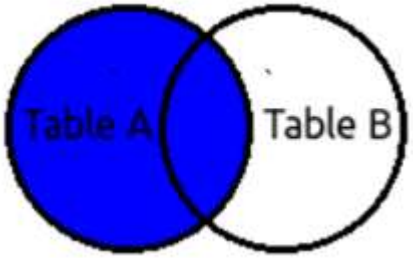
- **Right outer join:** keeps every tuple in S in the result of R S.

- **Full outer join:** keeps all tuples <u>in both the left and the right relations.</u> It is denoted by ⟗

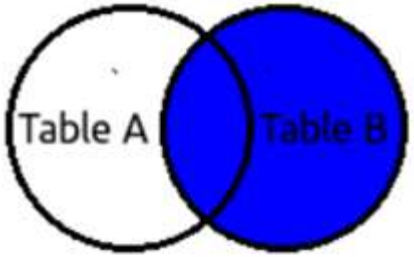
Inner and Outer Joins



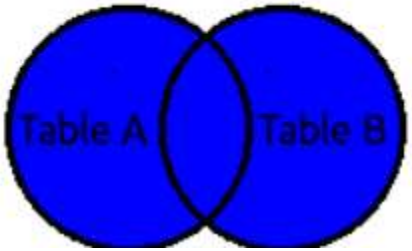
$\bowtie$



$\bowtie_{\supset}$



$\bowtie_{\subset}$



$\bowtie_{\supset\subset}$

Class Exercise - Another Example Outer Join

List the employees name and the Project name that they work on. If they don't work on any project or a project have no employee working on it, then indicate this with a null value.

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

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

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

DIFFERENT JOINS IN SQL

```
SELECT   E.FNAME, E.LNAME, S.FNAME, S.LNAME
FROM     EMPLOYEE AS E , EMPLOYEE AS S
WHERE    E.SUPERSSN=S.SSN
```

```
SELECT   E.FNAME, E.LNAME, S.FNAME, S.LNAME
FROM     (EMPLOYEE  E  JOIN EMPLOYEE  S)
             ON  E.SUPERSSN=S.SSN
```

```
SELECT   E.FNAME, E.LNAME, S.FNAME, S.LNAME
FROM     (EMPLOYEE  E LEFT OUTER JOIN EMPLOYEE  S
             ON  E.SUPERSSN=S.SSN)
```

NATURAL JOIN IN SQL

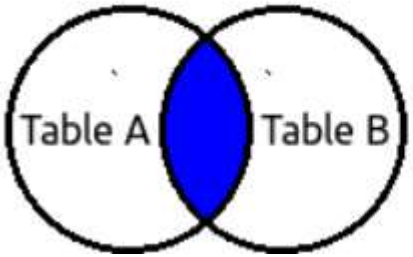
No Natural Join in **Transact-SQL**

The keyword **OUTER** is marked as optional that is

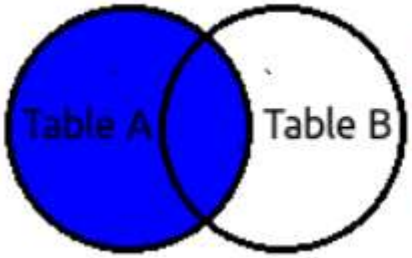
A LEFT JOIN B is same as A LEFT OUTER JOIN B

CROSS JOIN is for cartesian product

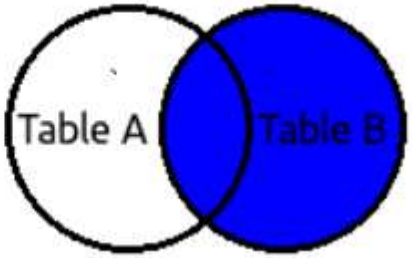
Inner and Outer Joins



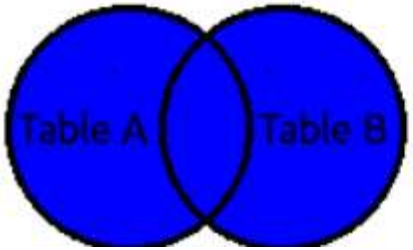

```
SELECT [list] FROM
    [Table A] A
INNER JOIN
    [Table B] B
ON A.Value = B.Value
```




```
SELECT [list] FROM
    [Table A] A
LEFT JOIN
    [Table B] B
ON A.Value = B.Value
```




```
SELECT [list] FROM
    [Table A] A
RIGHT JOIN
    [Table B] B
ON A.Value = B.Value
```




```
SELECT [list] FROM
    [Table A] A
FULL OUTER JOIN
    [Table B] B
ON A.Value = B.Value
```

General form of Grouping and Aggregation

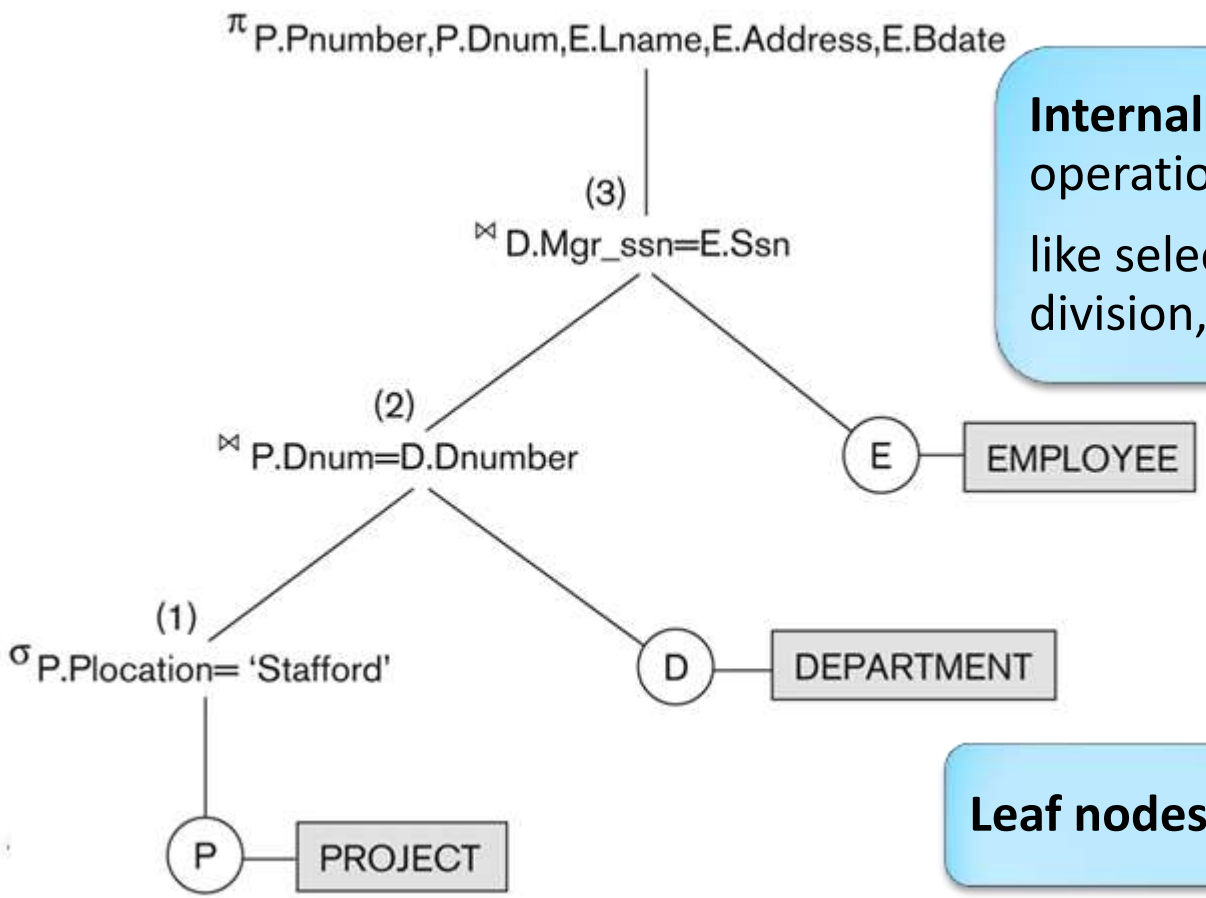
```
SELECT    S
FROM      R1,…,Rn
WHERE     C1
GROUP BY  a1,…,ak
HAVING    C2
```

Evaluation steps:

- Evaluate FROM-WHERE, apply condition C1
- Group by the attributes $a_1,\ldots,a_k$
- Apply condition C2 to each group (may have aggregates)
- Compute aggregates in S and return the result

Example of Query Tree

For every project located in **'Stafford'**, list the **project no**, the **controlling department no**, and the **department manager's last name, address, and birth date**.



Internal Nodes stand for operations

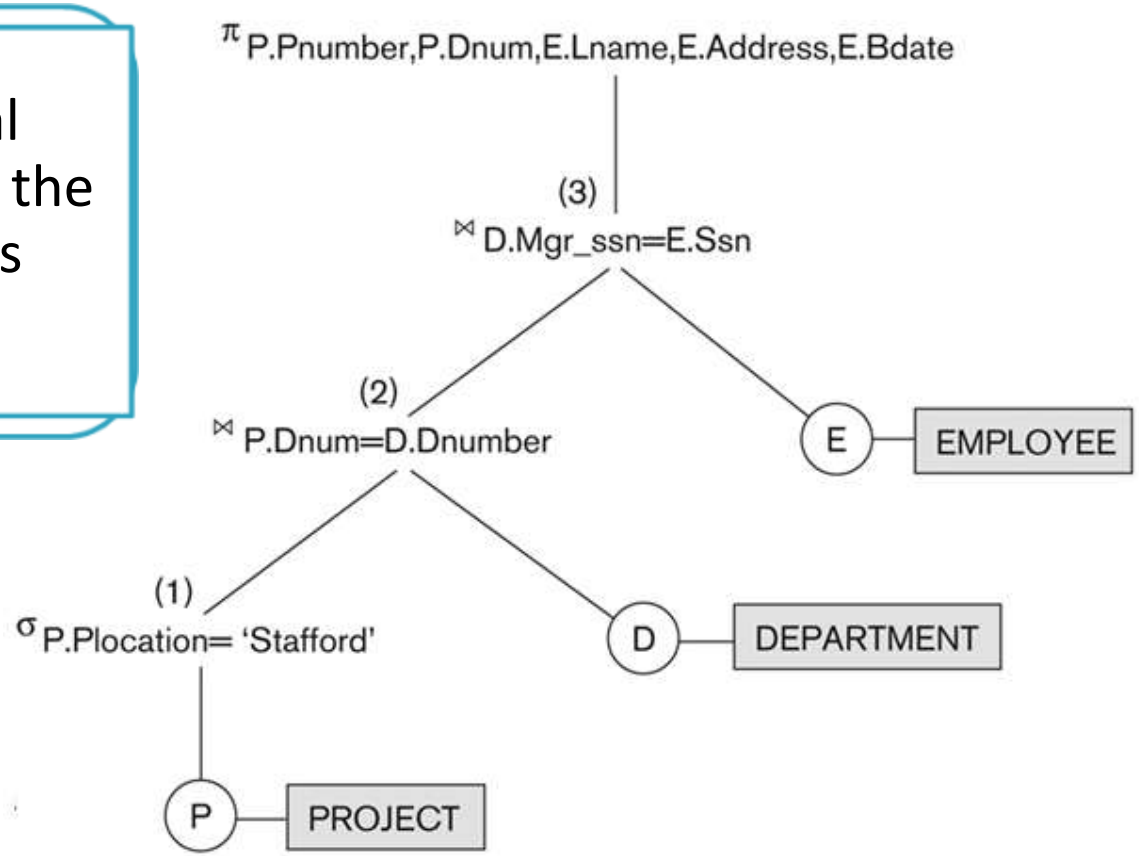
like selection, projection, join, division, ….

Leaf nodes represent **base relations**

Query Tree is an internal data structure to represent a query

Standard technique to **<u>estimate the work done</u>** in executing the query, and the ***optimization of execution***

A tree gives a good visual feel of the complexity of the query and the operations involved



Practice.. Practice.. Practice

Display the names of the departments that have no female employees.

```
Select Dname From Department
Except
Select Dname From Department join Employee on Dnumber = Dnum
Where Gender = 'F'
```

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

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

Practice.. Practice.. Practice

Find the SSN of all employees who are older than their Department Manager.

```
Select ssn
From (Employee as E join Department as D on E.dno=D.dnumber) join
Employee as M on mgr_ssn=ssn
Where E.birthdate < M.birthdate
```

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

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

Practice.. Practice.. Practice

Display the SSN and name of all employees who report to John. (in other words John is their immediate supervisor)

```
Select E.ssn
From Employee E join Employee S on E.Super_ssn=S.ssn and
Where S.Fname='John'
```

EMPLOYEE

FNAME	MINIT	LNAME	SSN	BDATE	ADDRESS	SEX	SALARY	SUPERSSN	DNO

DEPARTMENT

DNAME	DNUMBER	MGRSSN	MGRSTARTDATE

PROJECT

PNAME	PNUMBER	PLOCATION	DNUM

DEPT_LOCATIONS

DNUMBER	DLOCATION

WORKS_ON

ESSN	PNO	HOURS

Practice.. Practice.. Practice

Find out how many managers there are without listing them.

```
Select count(distinct mgr_ssn)
From department
```

EMPLOYEE

FNAME	MINIT	LNAME	SSN	BDATE	ADDRESS	SEX	SALARY	SUPERSSN	DNO

DEPARTMENT

DNAME	DNUMBER	MGRSSN	MGRSTARTDATE

PROJECT

PNAME	PNUMBER	PLOCATION	DNUM

DEPT_LOCATIONS

DNUMBER	DLOCATION

WORKS_ON

ESSN	PNO	HOURS

Practice.. Practice.. Practice

Find out the difference between highest and lowest salaries.

```
Select max(salary) - min(salary)
From department
```

EMPLOYEE

FNAME	MINIT	LNAME	SSN	BDATE	ADDRESS	SEX	SALARY	SUPERSSN	DNO

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

Practice.. Practice.. Practice

List SSN and Fname of all employees with more than 2 children

```
Select ssn, Fname
From employee join dependent on ssn=essn
Where Relationship = 'Son' or Relationship = 'daughter'
Groupby ssn
Having count(ssn)> 2
```

EMPLOYEE

FNAME	MINIT	LNAME	SSN	BDATE	ADDRESS	SEX	SALARY	SUPERSSN	DNO

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

Practice.. Practice.. Practice

- List the names of the departments, where all the employees have salary > 30000

```
SELECT Dname
FROM    Employee, Department
WHERE  dno= dnumber
GROUP BY  Dnumber, Dname
HAVING 30000 < min(Salary)
```

Example: Boat Rental database

- **Consider the following Boat Rental database schema:**
  - SAILOR (SID, SName, Phone, City)
  - BOAT (BName, BType, Price, OID)
  - RESERVATION (SID, BName, Date, Duration)
  - OWNER (OID, OName, Phone, Street, City, Country)

What does the query do?

```
SELECT Bname
FROM   (Boat b join Owner o on b.OID = o.OID)
         join Reservation r on r.BName = b.BName
WHERE  Country = 'Pakistan'
ORDER BY Price
```

Example: Boat Rental database

- **Consider the following Boat Rental database schema:**
  - SAILOR (<u>SID</u>, SName, Phone, City)
  - BOAT (<u>BName</u>, BType, Price, OID)
  - RESERVATION (<u>SID, BName</u>, Date, Duration)
  - OWNER (<u>OID</u>, OName, Phone, Street, City, Country)

- Select bname,count(*)
- From reservation r ,boat b,owner o
- Where b.bname=r.bname and b. oid=o.oid and country='USA'
- Group by bname
- Having count(*) > 10

What does the above query do?

Example: Boat Rental database

- Consider the following schema
  - SAILOR (<u>SID</u>, SName, Phone, City)
  - BOAT (<u>BName</u>, BType, Price, OID)
  - RESERVATION (<u>SID, BName</u>, Date, Duration)
  - OWNER (<u>OID</u>, OName, Phone, Street, City, Country)

- **Find the names of boats that are reserved by at least ten different sailors.**
- Select bname

  From reservation r

  Group by bname

  Having count(DISTINCT SID) >9

Example: Boat Rental database

- Consider the following schema
  - SAILOR (<u>SID</u>, SName, Phone, City)
  - BOAT (<u>BName</u>, BType, Price, OID)
  - RESERVATION (<u>SID, BName</u>, Date, Duration)
  - OWNER (<u>OID</u>, OName, Phone, Street, City, Country)

- **List name and price of the boats that were reserved in 2018 or in 2019.**

Select distinct b.bname, b.price
From reservation r join boat b on r.bname = b.bname
Where r.date LIKE '%2018%' or r.date LIKE '%2019%'

Example: Boat Rental database

- Consider the following schema
  - SAILOR (<u>SID</u>, SName, Phone, City)
  - BOAT (<u>BName</u>, BType, Price, OID)
  - RESERVATION (<u>SID, BName</u>, Date, Duration)
  - OWNER (<u>OID</u>, OName, Phone, Street, City, Country)
- **List name and price of the boats that were reserved in 2018 and in 2019.**

```
Select distinct b.bname, b.price
From reservation r, boat b
Where r.bname = b.bname and r.date LIKE '%2018%'
INTERSECT
Select distinct b.bname, b.price
From reservation r, boat b
Where r.bname = b.bname and r.date LIKE '%2019%'
```

Example: Boat Rental database

- Consider the following schema
  - SAILOR (<u>SID</u>, SName, Phone, City)
  - BOAT (<u>BName</u>, BType, Price, OID)
  - RESERVATION (<u>SID, BName</u>, Date, Duration)
  - OWNER (<u>OID</u>, OName, Phone, Street, City, Country)
- **List name, owner name, and price of the boats which were reserved in 2018 but not in 2019.**

```
Select distinct b.bname, b.price, o.oname
From reservation r, boat b, owner o
Where r.bname = b.bname and b.oid=o.oid and r.date LIKE '%2018%'
EXCEPT
Select distinct b.bname, b.price, o.oname
From reservation r, boat b, owner o
Where r.bname = b.bname and b.oid=o.oid and r.date LIKE '%2019%'
```

Boat Rental

The boat rental schema

- SAILOR (SID, SName, Phone, City)
- BOAT (BName, BType, Price, OID)
- RESERVATION (SID, BName, Date, Duration)
- OWNER (OID, OName, Phone, Street, City, Country)

List name and price of the boats which were reserved in 2018 and 2019 but not in 2020.

```
(Select distinct b.bname, b.price
From reservation r join boat b on r.bname = b.bname
Where r.date LIKE '%2018%'
INTERSECT
Select distinct b.bname, b.price
From reservation r join boat b on r.bname = b.bname
Where r.date LIKE '%2019%')
EXCEPT
Select distinct b.bname, b.price
From reservation r join boat b on r.bname = b.bname
Where r.date LIKE '%2020%'
```

Example: Boat Rental database

- Consider the following schema
  - SAILOR (<u>SID</u>, SName, Phone, City)
  - BOAT (<u>BName</u>, BType, Price, OID)
  - RESERVATION (<u>SID, BName</u>, Date, Duration)
  - OWNER (<u>OID</u>, OName, Phone, Street, City, Country)
- **Find ids of the sailors who only reserved a boat owned by Mr. Jonas with OID=12345**
  - All sailors who reserved a boat – sailors who have reserved a boat not owned by MR Jonas
- **Find ids of the sailors who have never reserved a boat owned by Mr. Jonas with OID=12345**
  - All sailors – sailors who have reserved a boat owned by MR jonas

SQL SERVER (TSQL) Functions

- Visit following slides for details on SQL functions
  - Like aggregate
  - String Functions
  - Date Functions
  - Math Functions

- https://docs.microsoft.com/en-us/sql/t-sql/functions/functions?view=sql-server-ver15

SQL QUERIES

Retrieve the name of all employees who earn more than the Average Salary

```
SELECT    FNAME, LNAME
FROM      EMPLOYEE
WHERE     Salary > AVG(SALARY)
```

DEPARTMENT

DNAME	DNUMBER	MGRSSN	MGRSTARTDATE

EMPLOYEE

FNAME	MINIT	LNAME	SSN	BDATE	ADDRESS	SEX	SALARY	SUPERSSN	DNO

NESTED QUERIES

A complete SELECT query, called a ***nested query***, can be specified within the WHERE-clause of another query, called the *outer query*

Retrieve the name of all employees who earn more than the Average Salary

```
SELECT    FNAME, LNAME
FROM      EMPLOYEE
WHERE     Salary > (SELECT  AVG(SALARY)
                    FROM    EMPLOYEE)
```

DEPARTMENT

DNAME	DNUMBER	MGRSSN	MGRSTARTDATE

EMPLOYEE

FNAME	MINIT	LNAME	SSN	BDATE	ADDRESS	SEX	SALARY	SUPERSSN	DNO

NESTING OF QUERIES

A complete SELECT query, called a ***nested query***, can be specified within the WHERE-clause of another query

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

```
SELECT    FNAME, LNAME
FROM      EMPLOYEE
WHERE     DNO = (SELECT DNUMBER
                 FROM   DEPARTMENT
                 WHERE  DNAME='Research' )
```

	FNAME	LNAME
1	John	Smith
2	Franklin	Wong
3	Joyce	English
4	Ramesh	Narayan

	DNUMBER
1	5

If `=` is used the inner query must return one value
If more than one value is returned then an error msg is generated

DEPARTMENT

DNAME	DNUMBER	MGRSSN	MGRSTARTDATE

EMPLOYEE

FNAME	MINIT	LNAME	SSN	BDATE	ADDRESS	SEX	SALARY	SUPERSSN	DNO

NESTING OF QUERIES

Retrieve the name of all employees who work for the 'Research' or Administration 'department .

```
SELECT    FNAME, LNAME
FROM      EMPLOYEE
WHERE     DNO IN  (SELECT  DNUMBER
                   FROM DEPARTMENT
                   WHERE DNAME='Research'  OR
                                 DNAME='Administration' )
```

If `=` is used the inner query must return one value.

If inner query returns more than one value then **use IN**

DEPARTMENT

DNAME	DNUMBER	MGRSSN	MGRSTARTDATE

EMPLOYEE

FNAME	MINIT	LNAME	SSN	BDATE	ADDRESS	SEX	SALARY	SUPERSSN	DNO

NESTING OF QUERIES

Retrieve the name of all employees who do not work for the 'Research' department.

```
SELECT   FNAME, LNAME
FROM     EMPLOYEE
WHERE    DNO NOT IN  (SELECT  DNUMBER
                      FROM DEPARTMENT
                      WHERE  DNAME='Research' )
```

DEPARTMENT

DNAME	DNUMBER	MGRSSN	MGRSTARTDATE

EMPLOYEE

FNAME	MINIT	LNAME	SSN	BDATE	ADDRESS	SEX	SALARY	SUPERSSN	DNO

NESTED QUERIES

You can also use: s > ALL R (means greater than every value)
s > ANY R (means greater than any value)

= ANY is same as IN , <>ALL is same as NOT IN

Find name of employees whose salary is greater than the salary of all employees in department 5

EMPLOYEE

FNAME	MINIT	LNAME	SSN	BDATE	ADDRESS	SEX	SALARY	SUPERSSN	DNO

```
SELECT  Fname
FROM    Employee
WHERE  Salary >  ALL (SELECT Salary
                      FROM    Employee
                      where Dno=5)
```

	Fname
1	James
2	Jennifer

NESTED QUERIES & TSQL

A subquery can be nested inside the **WHERE or HAVING** clause of an outer SELECT, INSERT, UPDATE, or DELETE statement.

Statements with subquery usually take one of these formats:

- WHERE expression [NOT] IN (subquery)
- WHERE expression **comparison_operator** [ANY | ALL] (subquery)
- WHERE [NOT] EXISTS (subquery)

comparison_operator { = | <> | != | > | >= | !> | < | <= | !< }

- Up to **32 levels** of nesting is possible,
  - This limit depends on available memory and the complexity of other expressions in the query.

Why NESTED QUERIES ?

- Many Transact-SQL statements that include subqueries can be alternatively formulated as joins.

- **Other questions can be posed only with subqueries.**

- An aggregate may not appear in the **WHERE clause**
  - unless it is in a subquery contained in a HAVING clause or a select list, and the column being aggregated is an outer reference

CORRELATED NESTED QUERIES

- If a condition in the ***nested query* references** an attribute of a relation **declared in the *outer query* =>**
  - **Then two queries are said to be <u>correlated</u>**

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

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 Correlated query is evaluated once for each tuple in outer query

CORRELATED NESTED QUERIES

A **correlated subquery** (*also called repeating subquery*) depends on the outer query for its values.

- This means that the nested subquery is executed **repeatedly**, once for each row that might be selected by the *outer query*.

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 SSN = ESSN AND FNAME=DEPENDENT_NAME)
```

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

- 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.

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 E, DEPENDENT D
WHERE     E.SSN=D.ESSN AND
          E.FNAME=D.DEPENDENT_NAME
```

EXISTS FUNCTION

EXISTS Function checks whether the result of a nested query is empty or not

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

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

EXISTS FUNCTION

Retrieve the names of employees who have no dependents.

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

EXISTS is necessary for the expressive power of SQL

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									

The above correlated nested query retrieves all DEPENDENT tuples related to an EMPLOYEE tuple.
If *none exist* , the EMPLOYEE tuple is selected

EXISTS FUNCTION

Find the names of managers who have at least one dependents.

```
SELECT   FNAME, LNAME
FROM     EMPLOYEE
WHERE    EXISTS   (SELECT  *
                   FROM  DEPARTMENT
                   WHERE SSN=Mgr_SSN)
         AND
         EXISTS   (SELECT  *
                   FROM  DEPENDENT
                   WHERE SSN=ESSN)
```

DEPARTMENT

DNAME	DNUMBER	MGRSSN	MGRSTARTDATE

DEPENDENT

Essn	Dependent_name	Sex	Bdate	Relationship

EMPLOYEE

FNAME	MINIT	LNAME	SSN	BDATE	ADDRESS	SEX	SALARY	SUPERSSN	DNO

EXISTS FUNCTION

Retrieve the name of each employee who works on *all* the projects controlled by department number 4.

Set theory: S1 contains S2 if (S2 – S1 = 0)

S1 = set of projects of each employee

S2 = set of Dept 4 projects

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

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

EXISTS FUNCTION

Retrieve the name of each employee who works on *all* the projects controlled by department number 4.

Set theory: S1 contains S2 if (S2 – S1 = 0)

```
SELECT    FNAME, LNAME
FROM      EMPLOYEE
WHERE     NOT EXISTS (
            (SELECT  PNUMBER
             FROM PROJECT
             WHERE     DNUM=4)

            EXCEPT

            (SELECT      PNO
             FROM        WORKS_ON
             WHERE       SSN=ESSN)
)
```

S1 = set of projects of each employee

S2 = set of dept 4 projects

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

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

HOW to do this in Relational Algebra?

Yet another Example

Find SSN of employees who work on all the projects of Dnum= 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

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

- PD4(Pno) $\leftarrow \pi_{\text{Pnumber}} (\sigma_{\text{DNUM=4}} \text{Project})$
- Ssn_Pnos $\leftarrow \pi_{\text{Essn,Pno}}$ (Works_on)
- SSNS(ssn)$\leftarrow$ Ssn_Pnos ??? PD4

DIVISION

Yet an other Example

Find SSN of employees who work on all the projects of Dnum= 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

PD4

Pno
10
30

SSN_PNOS

Essn	Pno
123456789	1
123456789	2
666884444	3
453453453	1
453453453	2
333445555	2
333445555	3
333445555	10
333445555	20
999887777	30
999887777	10
987987987	10
987987987	30
987654321	30
987654321	20
888665555	20

- PD4(Pno) ← $\pi_{\text{Pnumber}}(\sigma_{\text{DNUM=4}}\text{ Project})$
- Ssn_Pnos ← $\pi_{\text{Essn,Pno}}$ (Works_on)
- **SSNS(ssn)← Ssn_Pnos ÷ PD4**

DIVISION

DIVISION (Binary Operation)

Division operation is applied to two relations R1 and R2

R1(Attributes_R1) ÷ R2(Attributes_R2)

- where Attributes_R2 ⊂ Attributes_R1.

Let **Result = R1 ÷ R2**

Attr_Res = Attributes_R1 - Attributes_R2

- Attr_Res is a set of attributes of R1 that are not the attributes of R2.

R2

A
a1
a2
a3

Result

B
b1
b4

R1

A	B
a1	b1
a2	b1
a3	b1
a4	b1
a1	b2
a3	b2
a2	b3
a3	b3
a4	b3
a1	b4
a2	b4
a3	b4

For a **tuple t** to appear in the result of the DIVISION, the values in t must appear in R1 in combination with *every* tuple in R2.

Example of DIVISION

Find **SSN** of employees who work on all the projects that *John Smith* works on

EMPLOYEE

FNAME	MINIT	LNAME	SSN	BDATE	ADDRESS	SEX	SALARY	SUPERSSN	DNO

PROJECT

PNAME	PNUMBER	PLOCATION	DNUM

WORKS_ON

ESSN	PNO	HOURS

- Smith ← $\sigma_{\text{fname='John' and lname='Smith'}}$ (Employee)
- Smith_Pnos ← π_{Pno} (Works_on $\bowtie_{\text{essn=ssn}}$ Smith)
- Ssn_Pnos ← $\pi_{\text{Essn,Pno}}$ (Works_on)
- SSNS(ssn)← Ssn_Pnos ÷ Smith_Pnos

SMITH_PNOS

Pno
1
2

SSNS

Ssn
123456789
453453453

SSN_PNOS

Essn	Pno
123456789	1
123456789	2
666884444	3
453453453	1
453453453	2
333445555	2
333445555	3
333445555	10
333445555	20
999887777	30
999887777	10
987987987	10
987987987	30
987654321	30
987654321	20
888665555	20

Examples of Queries in RA

Find the names of employees who work on *all* the projects controlled by department number 5.

$T1(Pno) \leftarrow \pi_{Pnumber}(\sigma_{Dnum=5}(Project))$

$T2 \leftarrow \pi_{Essn, Pno}(Work\_On)$

$T3 \leftarrow (T2 \div T1)$

$R \leftarrow \pi_{LNAME, FNAME}(T3 * Employee)$

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

Essn	Pno
123456789	1
123456789	2
666884444	3
453453453	1
453453453	2
333445555	2
333445555	3
333445555	10
333445555	20
999887777	30
999887777	10
987987987	10
987987987	30
987654321	30
987654321	20
888665555	20

NESTED CORRELATED QUERIES

You can also use: s > ALL R
s > ANY R
EXISTS R

Find Employee whose salary is greater than the salary of all employee in department 5

```
SELECT  Fname
FROM    Employee
WHERE   Salary >  ALL (SELECT Salary
                       FROM    Employee
                       where Dno=5)
```

	Fname
1	James
2	Jennifer

Complex Correlated Query

Find Employees (dno and salary) whose salary is greater than the salaries of all employees in his department

```
SELECT   Fname, Salary, Dno
 FROM    Employee as E
 WHERE  Salary >  ALL (SELECT Salary
                       FROM    Employee as S
                       WHERE   E.dno=S.dno and E.ssn !=S.ssn )
```

	Fname	salary	Dno
1	Franklin	40000	5
2	James	55000	1
3	Jennifer	43000	4

Nested queries

- **Find the second highest salary**

```
SELECT MAX(Salary)
FROM Employee
WHERE Salary NOT IN (
        SELECT MAX(Salary)
        FROM Employee )
```

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

CORRELATED NESTED QUERIES

- **Find the third highest salary**

```
SELECT *
FROM Employee E1
WHERE (N-1) = (
        SELECT COUNT(DISTINCT(E2.Salary))
        FROM Employee E2
        WHERE E2.Salary > E1.Salary)
```

Find names of the departments such that all their employees have salary >30000

```
SELECT Dname
FROM    Employee, Department
WHERE  dno= dnumber
GROUP BY  Dnumber, Dname
HAVING 30000 < min(Salary)
```

Find names of the departments that have all employees with salary >30000

```
SELECT Dname
FROM    Department
WHERE  300000 < ALL (SELECT Salary
                     FROM Employee
                     WHERE dno= dnumber)
```

Almost equivalent…

```
SELECT Dname
FROM    Department
WHERE  Dnumber NOT IN (SELECT Dno
                       FROM Employee
                       WHERE  Salary<= 300000)
```

Group By and Having

- Count the number of employees whose salaries exceed **$30,000** in each department.
- Consider only the departments with more than five employees.

```
SELECT Dname, COUNT (*)
FROM DEPARTMENT, EMPLOYEE
WHERE Dnumber=Dno AND Salary>30000
GROUP BY Dname
HAVING COUNT (*) > 5;
```

Group By and Having

- Count the *total* number of employees whose salaries exceed $30,000 in each department, but only for departments where more than five employees work

- SELECT Dno, COUNT (*) No_of_Employees
  FROM EMPLOYEE
  WHERE salary > 30000 and DNO IN
  (SELECT Dno
  FROM EMPLOYEE
  GROUP BY Dno
  HAVING COUNT (*) > 5)
  Group by DNO

	Dno	No_of_Employees
1	4	1
2	5	2

Summary of SQL Queries

- A query in SQL can consist of up to six clauses, but only the first two, SELECT and FROM, are mandatory. The clauses are specified in the following order:

```
SELECT      <attribute list>
FROM        <table list>
[WHERE      <condition>]
[GROUP BY <grouping attribute(s)>]
[HAVING     <group condition>]
[ORDER BY <attribute list>]
```

- A query is evaluated by first applying the WHERE-clause, then GROUP BY and HAVING, and finally the SELECT-clause

https://docs.microsoft.com/en-us/sql/relational-databases/performance/subqueries?view=sql-server-2017

Performance of NESTED QUERIES in TSQL

In T-SQL, there is **usually** no performance difference between a statement that includes a subquery and a semantically equivalent version that does not.

In some cases where existence must be checked, a join yields better performance.

- Otherwise, the nested query must be processed for each result of the outer query to ensure elimination of duplicates. In such cases, a join approach would yield better results.

https://docs.microsoft.com/en-us/sql/relational-databases/performance/subqueries?view=sql-server-2017

SQL Queries

- There are various ways to specify the same query in SQL
  - This is to give flexibility to user to specify queries
- For query optimization, it is preferable to write a query with as little nesting and implied ordering as possible.
- Ideally, DBMS should process the same query in the same way regardless of how the query is specified.
  - But this is quite difficult in practice, (chapter 19,20)

Specifying Updates in SQL

There are three SQL commands to modify the database;

- INSERT,
- DELETE, and
- UPDATE

Example:

```
INSERT INTO  EMPLOYEE
VALUES ('Richard','K','Marini', '653298653', '30-DEC-52',
'98 Oak Forest,Katy,TX', 'M', 37000,'987654321', 4 )
```

INSERT WITH QUERY

- Suppose we want to create a temporary table that has the name, number of employees, and total salaries for each department.
- A table DEPTS_INFO is created by Q1, and is loaded with the information retrieved from the database by the query Q2.

- **Q1:**

```
CREATE TABLE  DEPTS_INFO
        (D_NAME          VARCHAR(10),
         NO_OF_EMPS      INTEGER,
         TOTAL_SAL       INTEGER);
```

- **Q2:**

```
INSERT INTO   DEPTS_INFO (D_NAME, NO_OF_EMPS, TOTAL_SAL)
        SELECT        DNAME, COUNT (*), SUM (SALARY)
        FROM          DEPARTMENT, EMPLOYEE
        WHERE         DNUMBER=DNO
        GROUP BY      DNAME ;
```

DELETE

- Removes tuples from a relation
- Tuples are deleted from only *one table* at a time (unless CASCADE is specified on a referential integrity constraint)
- <u>Examples:</u>

```
DELETE FROM    EMPLOYEE
WHERE          LNAME='Brown'


DELETE FROM    EMPLOYEE
WHERE          DNO      IN (SELECT  DNUMBER
                            FROM    DEPARTMENT
                            WHERE   DNAME='Research')


DELETE FROM    EMPLOYEE
```

UPDATE

- Used to modify attribute values of selected tuples
- **Example:** Change the location and controlling department number of project number 10 to 'Bellaire' and 5, respectively.

```
UPDATE PROJECT
SET    PLOCATION = 'Bellaire', DNUM = 5
WHERE  PNUMBER=10
```

PROJECT

PNAME	PNUMBER	PLOCATION	DNUM

UPDATE (cont.)

- **Example:** Give all employees in the 'Research' department a 10% raise in salary.

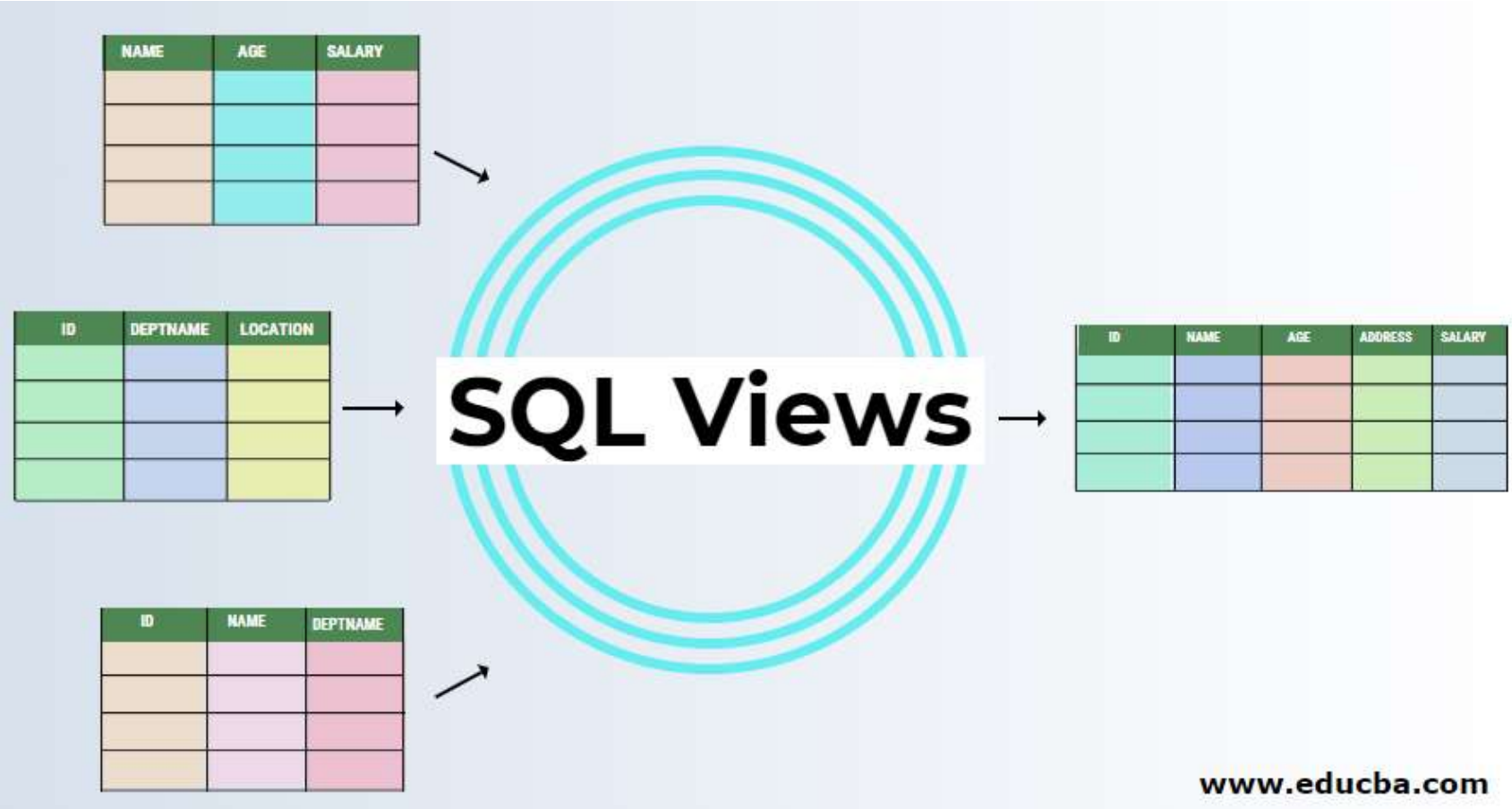
```
UPDATE EMPLOYEE
SET  SALARY = SALARY *1.1
WHERE  DNO  IN (SELECT  DNUMBER
                FROM DEPARTMENT
                WHERE DNAME='Research')
```

EMPLOYEE

FNAME	MINIT	LNAME	SSN	BDATE	ADDRESS	SEX	SALARY	SUPERSSN	DNO

Views in SQL

- A view is a **“virtual” table** that is derived from other tables



Views in SQL

- A view is a **"virtual" table** that is derived from other tables
- Allows for **limited update operations** (since the table may not physically be stored)
- **Allows full query operations**
- A convenience for expressing certain operations
  - simplify complex queries, and
  - define distinct conceptual interfaces for different users.

SQL Views: An Example

EMPLOYEE

FNAME	MINIT	LNAME	SSN	BDATE	ADDRESS	SEX	SALARY	SUPERSSN	DNO

PROJECT

PNAME	PNUMBER	PLOCATION	DNUM

WORKS_ON

ESSN	PNO	HOURS

CREATE VIEW WORKS_ON1 **AS**

SELECT FNAME, LNAME, PNAME, HOURS

FROM EMPLOYEE, PROJECT, WORKS_ON

WHERE SSN=ESSN AND PNO=PNUMBER

WORKS_ON1

Fname	Lname	Pname	Hours

SQL Views: An Example2

EMPLOYEE

FNAME	MINIT	LNAME	SSN	BDATE	ADDRESS	SEX	SALARY	SUPERSSN	DNO

DEPARTMENT

DNAME	DNUMBER	MGRSSN	MGRSTARTDATE

DEPT_INFO

Dept_name	No_of_emps	Total_sal

```
CREATE VIEW    DEPT_INFO(Dept_name, No_of_emps, Total_sal)
AS SELECT      Dname, COUNT (*), SUM (Salary)
    FROM       DEPARTMENT, EMPLOYEE
    WHERE      Dnumber=Dno
    GROUP BY   Dname;
```

Query using a Virtual Table

WORKS_ON1

Fname	Lname	Pname	Hours

- We can specify SQL queries on a newly created view:

```
SELECT FNAME, LNAME
FROM WORKS_ON1
WHERE PNAME='ProductX';
```

- DBMS is responsible to keep view always up-to-date
- When no longer needed, a view can be dropped:

```
DROP WORKS_ON1;
```

Efficient View Implementation

Query modification: present the view query in terms of a query on the underlying base tables

```
SELECT FNAME, LNAME
FROM WORKS_ON1
WHERE PNAME='ProductX'
```

WORKS_ON1

Fname	Lname	Pname	Hours

```
SELECT FNAME, LNAME
FROM (EMPLOYEE JOIN PROJECT on SSN=ESSN ) JOIN
                WORKS_ON on PNO=PNUMBER
WHERE PNAME='PRODUCTX'
```

Disadvantage:

Inefficient for views defined via complex queries

Esp. if additional queries are to be applied within a short time period

Efficient View Implementation

View materialization: involves physically creating and keeping a temporary table

- assumption: other queries on the view will follow
- concerns: maintaining correspondence between the base table and the view when the base table is updated
- strategy: incremental update

WORKS_ON1

Fname	Lname	Pname	Hours

View Update

Single view without aggregate operations:

- update may map to an update on the underlying base table

Views involving joins:

- an update *may* map to an update on the underlying base relations
- not always possible

EXAMPLE – Complex View Update

- **Example:**

  UPDATE WORKS_ON1

  SET PNAME=COMPUTERIZATION

  WHERE FNAME='JOHN AND

  LNAME='SMITH' AND

  PNAME='PRODUCTX'

WORKS_ON1

Fname	Lname	Pname	Hours

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

EXAMPLE – Complex View Update

UPDATE WORKS_ON1

SET PNAME=COMPUTERIZATION

WHERE FNAME='JOHN AND LNAME='SMITH' AND PNAME='PRODUCTX'

WORKS_ON1

Fname	Lname	Pname	Hours

A) UPDATE PROJECT
SET PNAME='COMPUTERIZATION'
WHERE PNAME='PRODUCTX'

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

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

View Update

```
UPDATE WORKS_ON1
SET PNAME=COMPUTERIZATION
WHERE  FNAME='JOHN  AND LNAME='SMITH' AND PNAME='PRODUCTX'
```

- B)**UPDATE WORKS_ON**

```
SET PNO =       (SELECT PNUMBER
                 FROM PROJECT
                 WHERE PNAME='COMPUTERIZATION')
WHERE ESSN IN          (SELECT SSN
                        FROM EMPLOYEE
                        WHERE   LNAME='SMITH' AND FNAME='JOHN')
                           AND
                           PNO = (SELECT PNUMBER FROM PROJECT
                                  WHERE PNAME='PRODUCTX')
```

Un-updatable Views

- Views defined using groups and aggregate functions are not updateable

```
UPDATE DEPT_INFO
SET    Total_sal=100000
WHERE  Dname='Research';
```

DEPT_INFO

Dept_name	No_of_emps	Total_sal

- Views defined on multiple tables using joins are generally not updateable

SQL Triggers

Triggers monitors a database and executes when an event occurs in the database server.

- like insertion,
- deletion or
- updation of data.

It is a database object which is bound to a table and is executed automatically.

You can’t explicitly invoke **trigger**s.

- The only way to do this is by performing the required action on the table that they are assigned to.

SQL Triggers

Objective: to monitor a database and take action when a condition occurs

Triggers include the following:

- event (e.g., an update operation)
- condition
- action (to be taken when the condition is satisfied)

Triggers are classified into two main types:

- After Triggers (For Triggers)
- Instead Of Triggers

SQL Triggers: An Example

Using a trigger with a reminder message

```
CREATE TRIGGER reminder1
ON Employee
AFTER INSERT, UPDATE
AS PRINT 'Notify employee added’
```

https://www.codeproject.com/Articles/25600/Triggers-SQL-Server

SQL Triggers: An Example

A trigger to compare an employee's salary to his/her supervisor after insert or update operations:

```
CREATE TRIGGER Emp_Salary ON Employee
    FOR INSERT, UPDATE
    AS
    IF EXISTS (SELECT * FROM inserted as i JOIN Employee as e ON
                    i.super_SSN= e.SSN WHERE i.salary > e.salary)
    BEGIN
        PRINT 'Employee salary is greater than the Supervisor Salary'
    END
```

```
INSERT INTO EMPLOYEE (FNAME, LNAME, SSN, Super_SSN, Salary)
VALUES ('Richard', 'Marini', '653298653','123456789',500000)
```

SQL Triggers

- **CREATE** TRIGGER SampleTrigger **ON** Employee
- **INSTEAD OF INSERT**
- **AS**
- SELECT * FROM Employee

- To fire the trigger we can insert a row in table and it will show list of all user instead of inserting into the table

```
INSERT INTO EMPLOYEE (FNAME, LNAME, SSN, Super_SSN, Salary)
VALUES ('Richard', 'Marini', '653298653','123456789',500000)
```

INSTEAD OF triggers are usually used to correctly update views that are based on multiple tables.

SQL Triggers: An Example

Using a trigger with a reminder message

```
CREATE TRIGGER reminder2
ON  employee
AFTER INSERT, UPDATE, DELETE
AS
        EXEC msdb.dbo.sp_send_dbmail
        @profile_name = ‘The Administrator’,
        @recipients = 'danw@Adventure-Works.com’,
        @body = 'Don''t forget to print a report’,
        @subject = 'Reminder’;
```

https://docs.microsoft.com/en-us/sql/t-sql/statements/create-trigger-transact-sql?view=sql-server-2017

Trigger

It is required that a team do not submit more than two proposals. Write a SQL query or trigger or view to solve this issue?

- INSTEAD OF triggers are run in place of the Insert command.
  - If you run insert command in instead of trigger it will again call the trigger so on.
- You can either use After (FOR) trigger
  - check if the inserted row has violated the given condition. If yes then delete it.

OR

- you can handle it at frontend application using Sql query to check if the given team has already submitted two projects then donot insert.

Why Transactions?

- *Transaction* is a process involving database queries and/or modification.
- Database systems are normally being accessed by many users or processes at the same time.
- Example- ATM
- Formed in SQL from single statements or explicit programmer control

ACID Transactions

Atomic	• Whole transaction or none is done.
Consistent	• Database constraints preserved.
Isolated	• It appears to the user as if only one process executes at a time.
Durable	• Effects of a process survive a crash.

Optional: weaker forms of transactions are often supported as well.

T-SQL AND Transactions

SQL has following transaction modes.

- Autocommit transactions
  - Each individual SQL statement = transaction.

- Explicit transactions

  BEGIN TRANSACTION

  [SQL statements]

  COMMIT or ROLLBACK

Transaction Support in TSQL

- BEGIN TRAN
- UPDATE Department
- SET Mgr_ssn = 123456789
- WHERE DNumber = 1
- UPDATE Department
- SET Mgr_start_date = '1981-06-19'
- WHERE Dnumber = 1
- COMMIT TRAN

Transaction Support in SQL

Potential problem with lower isolation levels:

- **Dirty Read**
  - Reading a value that was written by a failed transaction.

- **Nonrepeatable Read**
  - Allowing another transaction to write a new value between multiple reads of one transaction.
    - A transaction T1 reads a given value from a table.
    - If another transaction T2 later updates that value and T1 reads that value again, T1 will see a different value.

Transaction Support in SQL

- Potential problem with lower isolation levels (contd.):
  - Phantoms
    - New rows being read using the same read with a condition.
      - A transaction T1 may read a set of rows from a table, perhaps based on some condition specified in the SQL WHERE clause.
      - Now suppose that a transaction T2 inserts a new row that also satisfies the WHERE clause condition of T1, into the table used by T1.
      - If T1 is repeated, then T1 will see a row that previously did not exist, called a phantom.

Transaction Support in TSQL

Table 21.1 Possible Violations Based on Isolation Levels as Defined in SQL

	Type of Violation		
Isolation Level	**Dirty Read**	**Nonrepeatable Read**	**Phantom**
READ UNCOMMITTED	Yes	Yes	Yes
READ COMMITTED	No	Yes	Yes
REPEATABLE READ	No	No	Yes
SERIALIZABLE	No	No	No

Transaction Support in TSQL

1. "Dirty reads"

   SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED

2. "Committed reads"

   SET TRANSACTION ISOLATION LEVEL READ COMMITTED

3. "Repeatable reads"

   SET TRANSACTION ISOLATION LEVEL REPEATABLE READ

4. Serializable transactions (default):

   SET TRANSACTION ISOLATION LEVEL SERIALIZABLE

ACID Transactions

Atomic	• Whole transaction or none is done.
Consistent	• Database constraints preserved.
Isolated	• It appears to the user as if only one process executes at a time.
Durable	• Effects of a process survive a crash.

Optional: weaker forms of transactions are often supported as well.

Example of *Fund Transfer*

- Transaction to <u>transfer</u> $50 from account **A** to account **B**:
  1. **read**(*A*)
  2. *A* := *A* – 50
  3. **write**(*A*)
  4. **read**(*B*)
  5. *B* := *B* + 50
  6. **write**(*B*)

- **Atomicity requirement** :
  - if the transaction **fails** <u>after step 3</u> and <u>before step 6</u>,
    - the **system** should **ensure** that :
      - its **updates** are *not reflected* in the database,
      - <u>else</u> an *inconsistency* will result.

Example of *Fund Transfer*

- Transaction to <u>transfer</u> $50 from account **A** to account **B**:
  1. **read**(*A*)
  2. *A* := *A* – 50
  3. **write**(*A*)
  4. **read**(*B*)
  5. *B* := *B* + 50
  6. **write**(*B*)

- **Consistency requirement** :
  - the **sum** of **A** and **B** is:
    - **unchanged** <u>by</u> the <u>execution</u> of the transaction.

Example of *Fund Transfer* (Cont.)

- Transaction to transfer $50 from account **A** to account **B**:
  1. **read**(*A*)
  2. *A* := *A* – 50
  3. **write**(*A*)
  4. **read**(*B*)
  5. *B* := *B* + 50
  6. **write**(*B*)
- **Isolation requirement** —
  - if between steps 3 and 6, another transaction is allowed to access the partially updated database,
    - it will see an inconsistent database (the sum *A* + *B* will be less than it should be).
  - Isolation can be **ensured** trivially by:
    - running transactions **serially**, that is **one** after the **other**.
  - *However*, executing multiple *transactions* **concurrently** has significant **benefits**.

Example of *Fund Transfer* (Cont.)

- Transaction to transfer $50 from account **A** to account **B**:
  1. **read**(*A*)
  2. *A* := *A* – 50
  3. **write**(*A*)
  4. **read**(*B*)
  5. *B* := *B* + 50
  6. **write**(*B*)

- **Durability requirement** :
  - once the user has been notified that the transaction has **completed** :
    - (i.e., the transfer of the $50 has taken place),
    - the **updates** to the database by the transaction **must persist**
      - despite *failures*.