

NESTED QUERIES

You can also use $s > \text{ALL } R$
 $s > \text{ANY } R$
 $\text{EXISTS } R$

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	<u>Ssn</u>	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)
```

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

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

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

Almost equivalent...

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

```
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)**

	Dno	No_of_Employees
1	4	1
2	5	2

Group by DNO

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

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)
- Examples:

```
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	<u>PNUMBER</u>	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	<u>SSN</u>	BDATE	ADDRESS	SEX	SALARY	SUPERSSN	DNO
-------	-------	-------	------------	-------	---------	-----	--------	----------	-----

PRACTISE Problems

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)
- SELECT DISTINCT Bname
- FROM BOAT
- WHERE Price > ALL (SELECT price
FROM BOAT b , OWNER o
WHERE b.oid=o.oid and
Country='Pakistan')

What does the query do?



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)

- Select bname,count(*)
- From reservation r ,boat b,owner o
- Where b.bname=r.bname and b. oid=o.oid and country='Pakistan'
- Group by bname
- Having count(*) > 5

What does the above query do?



Example: Boat Rental database

- Consider the following schema
 - SAILOR (SID, SName, Phone, City)
 - BOAT (BName, BType, Price, OID)
 - RESERVATION (SID, BName, Date, Duration)
 - OWNER (OID, 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 (SID, SName, Phone, City)
 - BOAT (BName, BType, Price, OID)
 - RESERVATION (SID, BName, Date, Duration)
 - OWNER (OID, OName, Phone, Street, City, Country)
- List name, owner name, and price of the boats which were reserved in 2007 but not in 2008.

Select distinct b.bname, b.price, o.ename

From reservation r, boat b, owner o

Where r.bname = b.bname and b.oid=o.oid and r.date LIKE '%2007%'

EXCEPT

Select distinct b.bname, b.price, o.ename

From reservation r, boat b, owner o

Where r.bname = b.bname and b.oid=o.oid and r.date LIKE '%2008%'

EXAMPLE: BOAT RENTAL DATABASE

- Consider the following schema
 - SAILOR (SID, SName, Phone, City)
 - BOAT (BName, BType, Price, OID)
 - RESERVATION (SID, BName, Date, Duration)
 - OWNER (OID, OName, Phone, Street, City, Country)
- Find ids of sailors who have reserved all the boats of the owner with OID = 3459.
- Find pairs of the sailors who have reserved exactly same set of boats.

Example: Boat Rental database

- Consider the following schema
 - SAILOR (SID, SName, Phone, City)
 - BOAT (BName, BType, Price, OID)
 - RESERVATION (SID, BName, Date, Duration)
 - OWNER (OID, OName, Phone, Street, City, Country)
- Find ids of the sailors who **only** reserved a boat owned by Mr. Jonas with OID=12345
- Find ids of the sailors who have **never** reserved a boat owned by Mr. Jonas with OID=12345