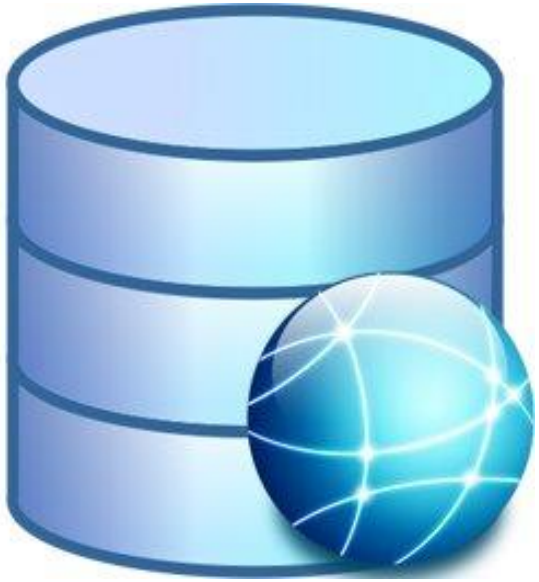
Database Systems

Structure Query Language (SQL)



Objectives

- Example Tables
- Introduction
- ISO SQL Data Types
- Comparison Operators in SQL
- Logical Operators in SQL
- Arithmetic Operators in SQL
- SQL Schema and Catalog
- SQL Data Definition Statements (DDL)
- SQL Data Manipulation Statements (DML)
- Other SQL Operators

Sample Tables

```
SQL> select * from dept;

    DEPTNO DNAME          LOC
---------- -------------- ---------
        10 ACCOUNTING     NEW YORK
        20 RESEARCH       DALLAS
        30 SALES          CHICAGO
        40 OPERATIONS     BOSTON
        50 MAINTENANCE    DALLAS

SQL> select * from emp;

     EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      7369 SMITH      CLERK           7902 17-DEC-80        800                    20
      7499 ALLEN      SALESMAN        7698 20-FEB-81       1600        300         30
      7521 WARD       SALESMAN        7698 22-FEB-81       1250        500         30
      7566 JONES      MANAGER         7839 02-APR-81       2975                    20
      7654 MARTIN     SALESMAN        7698 28-SEP-81       1250       1400         30
      7698 BLAKE      MANAGER         7839 01-MAY-81       2850                    30
      7782 CLARK      MANAGER         7839 09-JUN-81       2450                    10
      7788 SCOTT      ANALYST         7566 19-APR-87       3000                    20
      7839 KING       PRESIDENT            17-NOV-81       5000                    10
      7844 TURNER     SALESMAN        7698 08-SEP-81       1500          0         30
      7876 ADAMS      CLERK           7788 23-MAY-87       1100                    20
      7900 JAMES      CLERK           7698 03-DEC-81        950                    30
      7902 FORD       ANALYST         7566 03-DEC-81       3000
      7934 MILLER     CLERK           7782 23-JAN-82       1300                    10
14 rows selected.
```

Objectives of SQL

- Ideally, database language should allow user to:
  - create the database and relation structures;
  - perform insertion, modification, deletion of data from relations;
  - perform simple and complex queries.
- Must perform these tasks with minimal user effort and command structure and syntax must be easy to learn.
- It must be portable.
- SQL does not contain flow control commands. These must be implemented using a programming or job-control language, or interactively by the decisions of the user.

Objectives of SQL …

- SQL is relatively easy to learn:
  - It is a non-procedural language - you specify *what* information you require, rather than *how* to get it.
  - It is essentially free-format.

- Can be used by a range of users including DBAs, management, application programmers, and other types of end users.

- An ISO standard now exists for SQL, making it both the formal and *de facto* standard language for relational databases.

Objectives of SQL …

- Consists of standard English words:

```
CREATE TABLE staff(
                        sno      VARCHAR(5),
                         lname    VARCHAR(15),
                         salary    NUMBER(7,2)
                      );
```

```
INSERT INTO staff
VALUES ('SG16', 'Brown', 8300);
```

```
SELECT sno, lname, salary
FROM staff
WHERE salary > 10000;
```

History of SQL …

- In 1974, D. Chamberlin (IBM San Jose Laboratory) defined language called 'Structured English Query Language' or SEQUEL.
- A revised version SEQUEL/2 was defined in 1976 but name was subsequently changed to SQL for legal reasons.
- Still pronounced 'see-quel', though official pronunciation is 's-q-l'.
- IBM subsequently produced a prototype DBMS called *System R*, based on SEQUEL/2.
- Roots of SQL, however, are in SQUARE (Specifying Queries as Relational Expressions), which predates System R project.

History of SQL …

- In late 70s, ORACLE appeared and was probably first commercial RDBMS based on SQL.
- In 1987, ANSI and ISO published an initial standard for SQL.
- In 1989, ISO published an addendum that defined an 'Integrity Enhancement Feature'.
- In 1992, first major revision to ISO standard occurred, referred to as SQL2 or SQL/92.

Importance of SQL …

- SQL has become part of application architectures such as IBM's Systems Application Architecture (SAA).
- It is strategic choice of many large and influential organizations (e.g. X/OPEN).
- SQL is Federal Information Processing Standard (FIPS) to which conformance is required for all sales of databases to American Government.

Importance of SQL …

- SQL Access Group trying to define enhancements that will support interoperability across disparate systems.

- SQL is used in other standards and even influences development of other standards as a definitional tool. Examples include:
  - ISO's Information Resource Directory System (IRDS) Standard
  - Remote Data Access (RDA) Standard.

Components of SQL

- A database language must have support for the components listed below. Most implementations of SQL support various components listed below:
  - Data Definition Language (DDL)
  - Interactive Data Manipulation Language (Interactive DML)
  - Embedded Data Manipulation Language (Embedded DML)
  - Views
  - Integrity and transaction control
  - Authorization & Security (DCL)
  - Catalog and dictionary facility.

Basic Guidelines for Writing SQL Statements …

- SQL statement consists of *reserved words* and *user-defined words*.
  - Reserved words are a fixed part of SQL and must be spelt exactly as required and cannot be split across lines.
  - User-defined words are made up by user and represent names of various database objects such as relations, columns, views.
- Most components of an SQL statement are *case insensitive*, except for literal character data.
- More readable with indentation and lineation:
  - Each clause should begin on a new line.
  - Start of a clause should line up with start of other clauses.
  - If clause has several parts, should each appear on a separate line and be indented under start of clause.

Basic Guidelines for Writing SQL Statements …

- Use extended form of BNF notation:
  - Upper case letters represent reserved words.
  - Lower case letters represent user-defined words.
  - | indicates a *choice* among alternatives.
  - Curly braces indicate a *required element*.
  - Square brackets indicate an *optional element*.
  - … indicates *optional repetition* (0 or more).
  - ALL SQL is case less

ISO SQL Data Types

ISO SQL data types.

Data type	*Declarations*			
character	CHAR,	VARCHAR		
bit	BIT,	BIT VARYING		
exact numeric	NUMERIC,	DECIMAL,	INTEGER,	SMALLINT
approximate numeric	FLOAT,	REAL,	DOUBLE PRECISION	
datetime	DATE,	TIME,	TIMESTAMP	
interval	INTERVAL			

Comparison Operators in SQL

- There are six comparison operators in SQL. These operators are used to build conditions that are used in the WHERE clause of a DML statement:

Operator	Meaning
=	Equal
<>	Not Equal
<	Less than
>	Greater than
<=	Less than or Eqaul
>=	Greater than or Eqaul

Logical Operators in SQL

- There are three logical operators that help us to build compound conditions to be used in the WHERE clause of the SELECT statement.
  - The **AND** operator joins two or more conditions, and display a row only if that row's data satisfies ALL the specified conditions.
  - The **OR** operator joins two or more conditions, and display a row only if that row's data satisfies any of the specified conditions.
  - The **NOT** is a unary operator, and is used to negates a condition.

Arithmetic Operators in SQL

- Another feature of SQL allows the use of arithmetic in queries.
  - The standard arithmetic operators (+, -, /, *) can be applied to numeric values or attributes with numeric domain.
  - The arithmetic operators can be used in expressions in the SELECT and the WHERE clauses to compute numeric values.
  - All attributes that can be computed using arithmetic expressions (such as age from birth date, annual salary from monthly salary) must be eliminated as part of a good design practice in databases.

SQL Schema and Catalog

- In SQL92, relations and other database objects exist in an **environment**.
- Each environment contains one or more **catalogs**, and each catalog consists of set of **schemas**.
- **Schema** is a named collection of related database objects.
- Objects in a schema can be tables, views, domains, constraints, translations, and character sets. All have same owner.

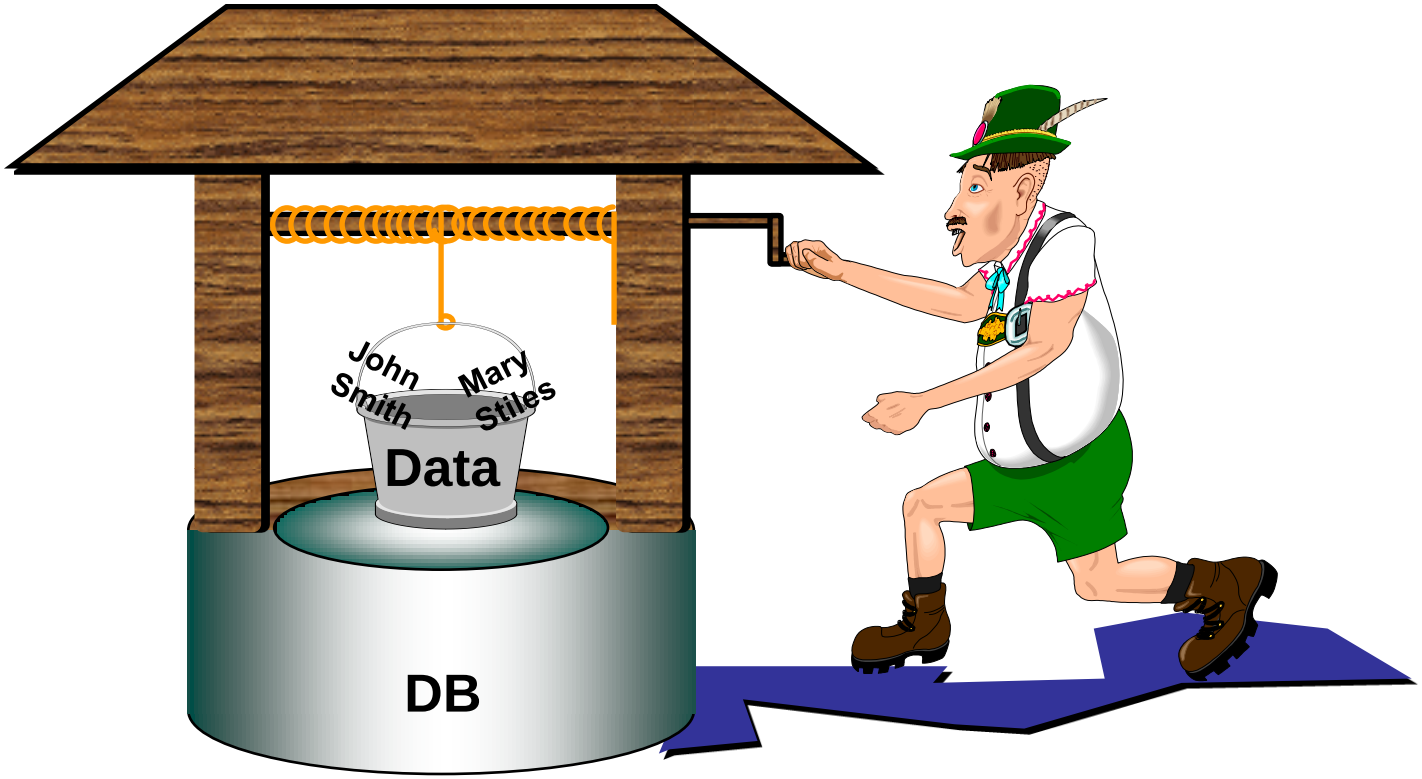
SELECT

- SELECT Definition
- Selecting Columns
- Selecting Rows
- Sorting
- Aggregation
- Grouping
- Restricting Groups
- Aliasing Table Names
- Nested Queries
- Join
- Set Operations

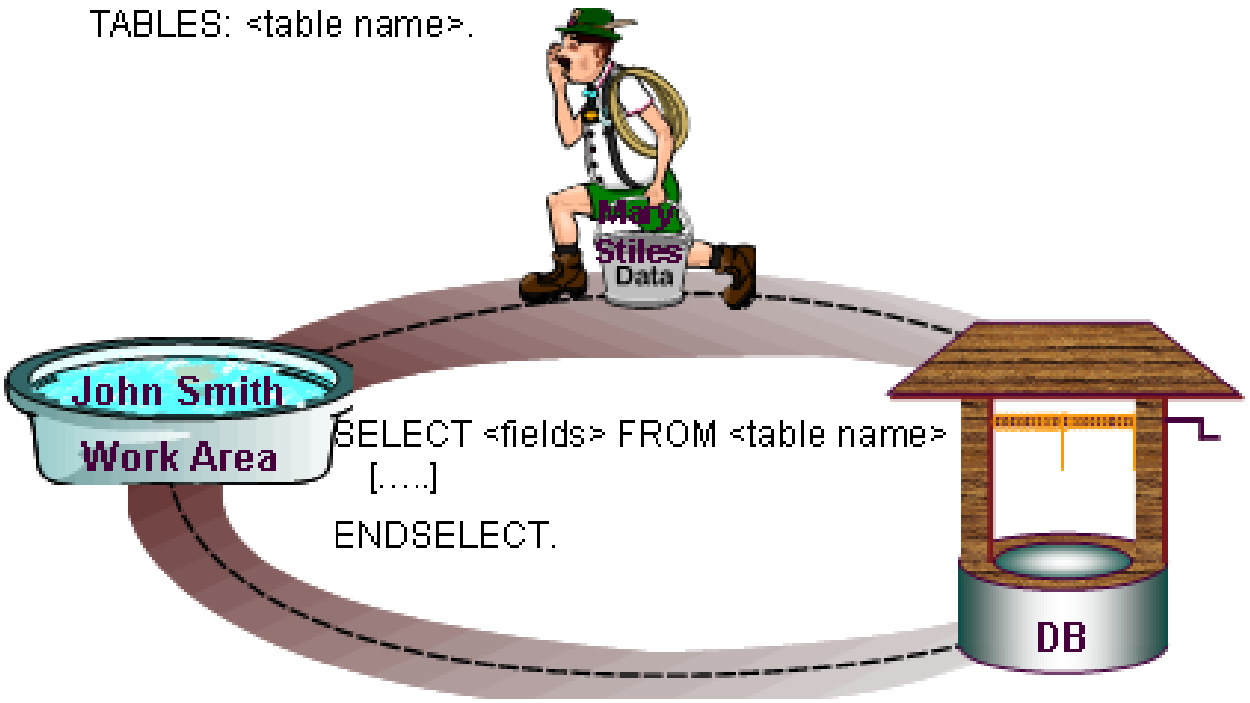
SELECT Definition …

- SQL has only one statement for retrieving information from a database called the SELECT statement.
- SQL SELECT statement is different from that of Relational Algebra.
- An important distinction between SQL and formal relational model is that SQL allows duplicate rows. Hence an SQL table is not a set but a multiset (some times called a bag) of tuples.

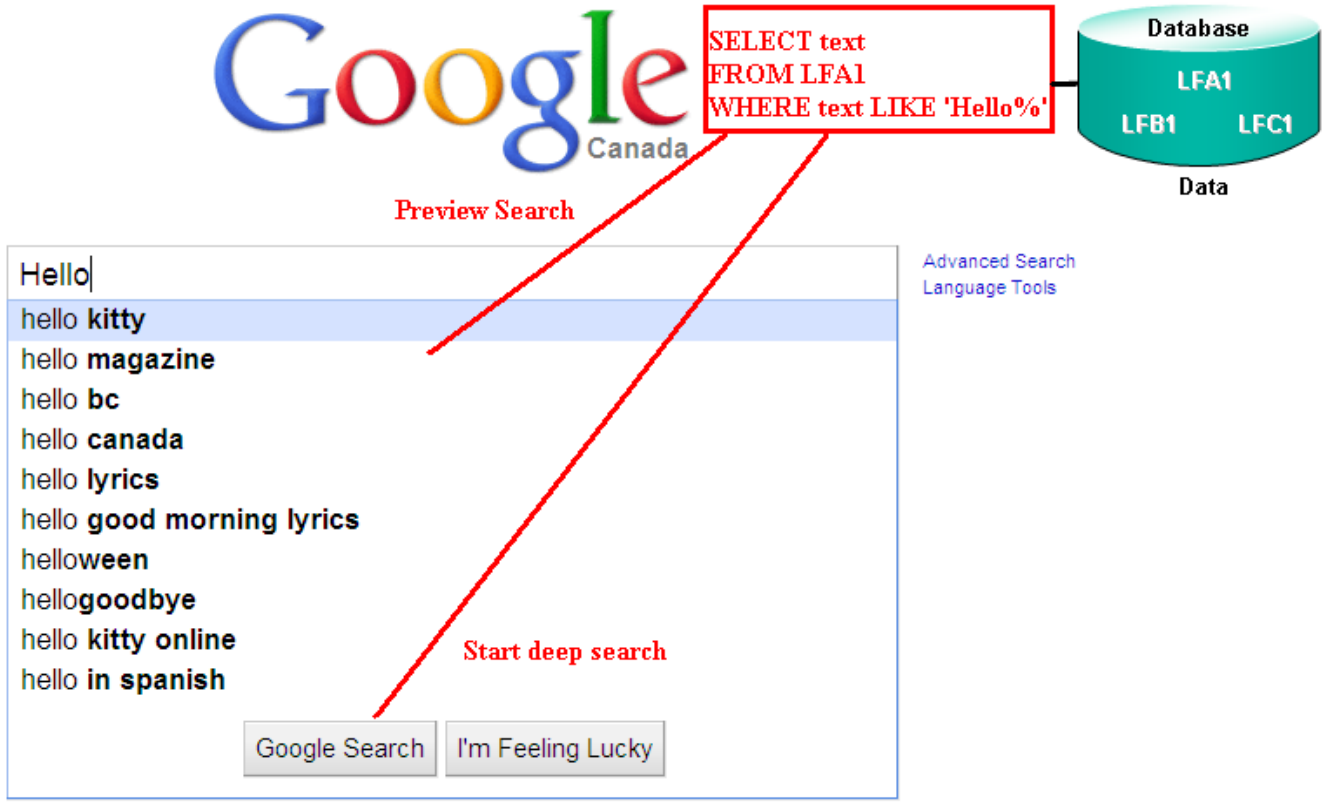
Retrieving Information From the Database



SQL


TABLES: <table name>.
Mary
Stiles
Data
John Smith
Work Area
SELECT <fields> FROM <table name>
[....]
ENDSELECT.
DB

SQL - Usage


Google
Canada
SELECT text
FROM LFA1
WHERE text LIKE 'Hello%'
Database
LFA1
LFB1
LFC1
Data
Preview Search
Hello
hello kitty
hello magazine
hello bc
hello canada
hello lyrics
hello good morning lyrics
helloween
hellogoodbye
hello kitty online
hello in spanish
Advanced Search
Language Tools
Start deep search
Google Search
I'm Feeling Lucky

SELECT Definition …

- A SELECT statement can consist up to six clauses.

```
SELECT      [DISTINCT | ALL]
              {*     |     [column_expression     [AS
  new_name]] [,...] }
FROM        table_name [alias] [, ...]
[WHERE      condition]
[GROUP BY     column_list]
[HAVING       condition]
[ORDER By     column_list]
```

- Only **SELECT** and **FROM** clauses are mandatory.
- Order of the clauses cannot be changed.

SELECT Definition …

- **FROM** Specifies table(s) to be used.
- **WHERE** Filters rows.
- **GROUP BY** Forms groups of rows with same column value.
- **HAVING** Filters groups subject to some condition.
- **SELECT** Specifies which columns are to appear in output.
- **ORDER BY** Specifies the order of the output.

Selecting Columns

- Selecting all columns +
- Selecting Specific columns +
- Selecting Computed columns +
- Renaming Columns +

Selecting ALL Columns

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	17-DEC-80	800		20
7499	ALLEN	SALESMAN	7698	20-FEB-81	1600	300	30
7521	WARD	SALESMAN	7698	22-FEB-81	1250	500	30
7566	JONES	MANAGER	7839	02-APR-81	2975		20
7654	MARTIN	SALESMAN	7698	28-SEP-81	1250	1400	30
7698	BLAKE	MANAGER	7839	01-MAY-81	2850		30
7782	CLARK	MANAGER	7839	09-JUN-81	2450		10
7788	SCOTT	ANALYST	7566	19-APR-87	3000		20
7839	KING	PRESIDENT		17-NOV-81	5000		10
7844	TURNER	SALESMAN	7698	08-SEP-81	1500	0	30
7876	ADAMS	CLERK	7788	23-MAY-87	1100		20
7900	JAMES	CLERK	7698	03-DEC-81	950		30
7902	FORD	ANALYST	7566	03-DEC-81	3000		
7934	MILLER	CLERK	7782	23-JAN-82	1300		10

```
SELECT *
FROM emp;
```

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	17-DEC-80	800		20
7499	ALLEN	SALESMAN	7698	20-FEB-81	1600	300	30
7521	WARD	SALESMAN	7698	22-FEB-81	1250	500	30
7566	JONES	MANAGER	7839	02-APR-81	2975		20
7654	MARTIN	SALESMAN	7698	28-SEP-81	1250	1400	30
7698	BLAKE	MANAGER	7839	01-MAY-81	2850		30
7782	CLARK	MANAGER	7839	09-JUN-81	2450		10
7788	SCOTT	ANALYST	7566	19-APR-87	3000		20
7839	KING	PRESIDENT		17-NOV-81	5000		10
7844	TURNER	SALESMAN	7698	08-SEP-81	1500	0	30
7876	ADAMS	CLERK	7788	23-MAY-87	1100		20
7900	JAMES	CLERK	7698	03-DEC-81	950		30
7902	FORD	ANALYST	7566	03-DEC-81	3000		
7934	MILLER	CLERK	7782	23-JAN-82	1300		10

Selecting Specific Columns

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	17-DEC-80	800		20
7499	ALLEN	SALESMAN	7698	20-FEB-81	1600	300	30
7521	WARD	SALESMAN	7698	22-FEB-81	1250	500	30
7566	JONES	MANAGER	7839	02-APR-81	2975		20
7654	MARTIN	SALESMAN	7698	28-SEP-81	1250	1400	30
7698	BLAKE	MANAGER	7839	01-MAY-81	2850		30
7782	CLARK	MANAGER	7839	09-JUN-81	2450		10
7788	SCOTT	ANALYST	7566	19-APR-87	3000		20
7839	KING	PRESIDENT		17-NOV-81	5000		10
7844	TURNER	SALESMAN	7698	08-SEP-81	1500	0	30
7876	ADAMS	CLERK	7788	23-MAY-87	1100		20
7900	JAMES	CLERK	7698	03-DEC-81	950		30
7902	FORD	ANALYST	7566	03-DEC-81	3000		
7934	MILLER	CLERK	7782	23-JAN-82	1300		10

```
SELECT ename, deptno
FROM emp;
```

ENAME	DEPTNO
SMITH	20
ALLEN	30
WARD	30
JONES	20
MARTIN	30
BLAKE	30
CLARK	10
SCOTT	20
KING	10
TURNER	30
ADAMS	20
JAMES	30
FORD	
MILLER	10

Selecting Computed Columns

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	17-DEC-80	800		20
7499	ALLEN	SALESMAN	7698	20-FEB-81	1600	300	30
7521	WARD	SALESMAN	7698	22-FEB-81	1250	500	30
7566	JONES	MANAGER	7839	02-APR-81	2975		20
7654	MARTIN	SALESMAN	7698	28-SEP-81	1250	1400	30
7698	BLAKE	MANAGER	7839	01-MAY-81	2850		30
7782	CLARK	MANAGER	7839	09-JUN-81	2450		10
7788	SCOTT	ANALYST	7566	19-APR-87	3000		20
7839	KING	PRESIDENT		17-NOV-81	5000		10
7844	TURNER	SALESMAN	7698	08-SEP-81	1500	0	30
7876	ADAMS	CLERK	7788	23-MAY-87	1100		20
7900	JAMES	CLERK	7698	03-DEC-81	950		30
7902	FORD	ANALYST	7566	03-DEC-81	3000		
7934	MILLER	CLERK	7782	23-JAN-82	1300		10

```
SELECT empno, ename, sal*12
FROM emp;
```

EMPNO	ENAME	SAL*12
7369	SMITH	9600
7499	ALLEN	19200
7521	WARD	15000
7566	JONES	35700
7654	MARTIN	15000
7698	BLAKE	34200
7782	CLARK	29400
7788	SCOTT	36000
7839	KING	60000
7844	TURNER	18000
7876	ADAMS	13200
7900	JAMES	11400
7902	FORD	36000
7934	MILLER	15600

Renaming Columns

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	17-DEC-80	800		20
7499	ALLEN	SALESMAN	7698	20-FEB-81	1600	300	30
7521	WARD	SALESMAN	7698	22-FEB-81	1250	500	30
7566	JONES	MANAGER	7839	02-APR-81	2975		20
7654	MARTIN	SALESMAN	7698	28-SEP-81	1250	1400	30
7698	BLAKE	MANAGER	7839	01-MAY-81	2850		30
7782	CLARK	MANAGER	7839	09-JUN-81	2450		10
7788	SCOTT	ANALYST	7566	19-APR-87	3000		20
7839	KING	PRESIDENT		17-NOV-81	5000		10
7844	TURNER	SALESMAN	7698	08-SEP-81	1500	0	30
7876	ADAMS	CLERK	7788	23-MAY-87	1100		20
7900	JAMES	CLERK	7698	03-DEC-81	950		30
7902	FORD	ANALYST	7566	03-DEC-81	3000		
7934	MILLER	CLERK	7782	23-JAN-82	1300		10

```
SELECT empno, ename, sal*12 salary
FROM emp;
```

EMPNO	ENAME	SALARY
7369	SMITH	9600
7499	ALLEN	19200
7521	WARD	15000
7566	JONES	35700
7654	MARTIN	15000
7698	BLAKE	34200
7782	CLARK	29400
7788	SCOTT	36000
7839	KING	60000
7844	TURNER	18000
7876	ADAMS	13200
7900	JAMES	11400
7902	FORD	36000
7934	MILLER	15600

Selecting Rows

- Selecting All Rows +
- Partial match Search +
- Range Search +
- Set Membership Search +
- Pattern matching Search +
- Null Search +
- Removing Duplicate Rows +

Selecting ALL Rows

```
SELECT *
FROM emp;
```

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	17-DEC-80	800		20
7499	ALLEN	SALESMAN	7698	20-FEB-81	1600	300	30
7521	WARD	SALESMAN	7698	22-FEB-81	1250	500	30
7566	JONES	MANAGER	7839	02-APR-81	2975		20
7654	MARTIN	SALESMAN	7698	28-SEP-81	1250	1400	30
7698	BLAKE	MANAGER	7839	01-MAY-81	2850		30
7782	CLARK	MANAGER	7839	09-JUN-81	2450		10
7788	SCOTT	ANALYST	7566	19-APR-87	3000		20
7839	KING	PRESIDENT		17-NOV-81	5000		10
7844	TURNER	SALESMAN	7698	08-SEP-81	1500	0	30
7876	ADAMS	CLERK	7788	23-MAY-87	1100		20
7900	JAMES	CLERK	7698	03-DEC-81	950		30
7902	FORD	ANALYST	7566	03-DEC-81	3000		
7934	MILLER	CLERK	7782	23-JAN-82	1300		10

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	17-DEC-80	800		20
7499	ALLEN	SALESMAN	7698	20-FEB-81	1600	300	30
7521	WARD	SALESMAN	7698	22-FEB-81	1250	500	30
7566	JONES	MANAGER	7839	02-APR-81	2975		20
7654	MARTIN	SALESMAN	7698	28-SEP-81	1250	1400	30
7698	BLAKE	MANAGER	7839	01-MAY-81	2850		30
7782	CLARK	MANAGER	7839	09-JUN-81	2450		10
7788	SCOTT	ANALYST	7566	19-APR-87	3000		20
7839	KING	PRESIDENT		17-NOV-81	5000		10
7844	TURNER	SALESMAN	7698	08-SEP-81	1500	0	30
7876	ADAMS	CLERK	7788	23-MAY-87	1100		20
7900	JAMES	CLERK	7698	03-DEC-81	950		30
7902	FORD	ANALYST	7566	03-DEC-81	3000		
7934	MILLER	CLERK	7782	23-JAN-82	1300		10

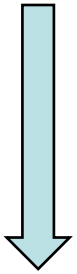
Selecting Rows

- To Select certain rows of a table you need to use the WHERE clause of the SELECT statement.
- The WHERE clause has a condition which is a logical expression.
- The Where condition consists of:
  - Comparison Operators
  - Logical Operators
  - Arithmetic Operators
  - Other SQL constructs which will be discussed later.
- A record to be selected it must make the WHERE logical expression true. In other words it must satisfy the where condition.

Partial Match Columns

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	17-DEC-80	800		20
7499	ALLEN	SALESMAN	7698	20-FEB-81	1600	300	30
7521	WARD	SALESMAN	7698	22-FEB-81	1250	500	30
7566	JONES	MANAGER	7839	02-APR-81	2975		20
7654	MARTIN	SALESMAN	7698	28-SEP-81	1250	1400	30
7698	BLAKE	MANAGER	7839	01-MAY-81	2850		30
7782	CLARK	MANAGER	7839	09-JUN-81	2450		10
7788	SCOTT	ANALYST	7566	19-APR-87	3000		20
7839	KING	PRESIDENT		17-NOV-81	5000		10
7844	TURNER	SALESMAN	7698	08-SEP-81	1500	0	30
7876	ADAMS	CLERK	7788	23-MAY-87	1100		20
7900	JAMES	CLERK	7698	03-DEC-81	950		30
7902	FORD	ANALYST	7566	03-DEC-81	3000		
7934	MILLER	CLERK	7782	23-JAN-82	1300		10

```
SELECT *
FROM emp
WHERE deptno=20;
```

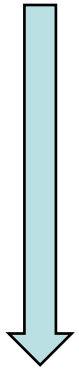


EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	17-DEC-80	800		20
7566	JONES	MANAGER	7839	02-APR-81	2975		20
7788	SCOTT	ANALYST	7566	19-APR-87	3000		20
7876	ADAMS	CLERK	7788	23-MAY-87	1100		20

Partial Match Columns …

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	17-DEC-80	800		20
7499	ALLEN	SALESMAN	7698	20-FEB-81	1600	300	30
7521	WARD	SALESMAN	7698	22-FEB-81	1250	500	30
7566	JONES	MANAGER	7839	02-APR-81	2975		20
7654	MARTIN	SALESMAN	7698	28-SEP-81	1250	1400	30
7698	BLAKE	MANAGER	7839	01-MAY-81	2850		30
7782	CLARK	MANAGER	7839	09-JUN-81	2450		10
7788	SCOTT	ANALYST	7566	19-APR-87	3000		20
7839	KING	PRESIDENT		17-NOV-81	5000		10
7844	TURNER	SALESMAN	7698	08-SEP-81	1500	0	30
7876	ADAMS	CLERK	7788	23-MAY-87	1100		20
7900	JAMES	CLERK	7698	03-DEC-81	950		30
7902	FORD	ANALYST	7566	03-DEC-81	3000		
7934	MILLER	CLERK	7782	23-JAN-82	1300		10

```
SELECT *
FROM emp
WHERE deptno=20
AND ename='CLERK';
```



EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	17-DEC-80	800		20
7876	ADAMS	CLERK	7788	23-MAY-87	1100		20

Range Search …

Selecting all the records whose column values is between the values specified in the WHERE clause.

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	17-DEC-80	800		20
7499	ALLEN	SALESMAN	7698	20-FEB-81	1600	300	30
7521	WARD	SALESMAN	7698	22-FEB-81	1250	500	30
7566	JONES	MANAGER	7839	02-APR-81	2975		20
7654	MARTIN	SALESMAN	7698	28-SEP-81	1250	1400	30
7698	BLAKE	MANAGER	7839	01-MAY-81	2850		30
7782	CLARK	MANAGER	7839	09-JUN-81	2450		10
7788	SCOTT	ANALYST	7566	19-APR-87	3000		20
7839	KING	PRESIDENT		17-NOV-81	5000		10
7844	TURNER	SALESMAN	7698	08-SEP-81	1500	0	30
7876	ADAMS	CLERK	7788	23-MAY-87	1100		20
7900	JAMES	CLERK	7698	03-DEC-81	950		30
7902	FORD	ANALYST	7566	03-DEC-81	3000		
7934	MILLER	CLERK	7782	23-JAN-82	1300		10

```
SELECT empno, ename, job, sal
FROM emp
WHERE sal >=800 AND sal <= 2000;
```

```
SELECT empno, ename, job, sal
FROM emp
WHERE sal BETWEEN 800 and 2000;
```

EMPNO	ENAME	JOB	SAL
7369	SMITH	CLERK	800
7499	ALLEN	SALESMAN	1600
7521	WARD	SALESMAN	1250
7654	MARTIN	SALESMAN	1250
7844	TURNER	SALESMAN	1500
7876	ADAMS	CLERK	1100
7900	JAMES	CLERK	950
7934	MILLER	CLERK	1300

Set Membership Search …

Selecting all the records whose column value is a member of the set specified in the WHERE clause.

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	17-DEC-80	800		20
7499	ALLEN	SALESMAN	7698	20-FEB-81	1600	300	30
7521	WARD	SALESMAN	7698	22-FEB-81	1250	500	30
7566	JONES	MANAGER	7839	02-APR-81	2975		20
7654	MARTIN	SALESMAN	7698	28-SEP-81	1250	1400	30
7698	BLAKE	MANAGER	7839	01-MAY-81	2850		30
7782	CLARK	MANAGER	7839	09-JUN-81	2450		10
7788	SCOTT	ANALYST	7566	19-APR-87	3000		20
7839	KING	PRESIDENT		17-NOV-81	5000		10
7844	TURNER	SALESMAN	7698	08-SEP-81	1500	0	30
7876	ADAMS	CLERK	7788	23-MAY-87	1100		20
7900	JAMES	CLERK	7698	03-DEC-81	950		30
7902	FORD	ANALYST	7566	03-DEC-81	3000		
7934	MILLER	CLERK	7782	23-JAN-82	1300		10

```
SELECT empno, ename, job, sal
FROM emp
WHERE job IN ('CLERK', 'MANAGER');
```

EMPNO	ENAME	JOB	SAL
7369	SMITH	CLERK	800
7566	JONES	MANAGER	2975
7698	BLAKE	MANAGER	2850
7782	CLARK	MANAGER	2450
7876	ADAMS	CLERK	1100
7900	JAMES	CLERK	950
7934	MILLER	CLERK	1300

Set Membership Search …

Selecting all the records whose column value is a member of the set specified in the WHERE clause.

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	17-DEC-80	800		20
7499	ALLEN	SALESMAN	7698	20-FEB-81	1600	300	30
7521	WARD	SALESMAN	7698	22-FEB-81	1250	500	30
7566	JONES	MANAGER	7839	02-APR-81	2975		20
7654	MARTIN	SALESMAN	7698	28-SEP-81	1250	1400	30
7698	BLAKE	MANAGER	7839	01-MAY-81	2850		30
7782	CLARK	MANAGER	7839	09-JUN-81	2450		10
7788	SCOTT	ANALYST	7566	19-APR-87	3000		20
7839	KING	PRESIDENT		17-NOV-81	5000		10
7844	TURNER	SALESMAN	7698	08-SEP-81	1500	0	30
7876	ADAMS	CLERK	7788	23-MAY-87	1100		20
7900	JAMES	CLERK	7698	03-DEC-81	950		30
7902	FORD	ANALYST	7566	03-DEC-81	3000		
7934	MILLER	CLERK	7782	23-JAN-82	1300		10

```
SELECT empno, ename, job, sal
FROM emp
WHERE job NOT IN ('CLERK', 'MANAGER');
```

EMPNO	ENAME	JOB	SAL
7499	ALLEN	SALESMAN	1600
7521	WARD	SALESMAN	1250
7654	MARTIN	SALESMAN	1250
7788	SCOTT	ANALYST	3000
7839	KING	PRESIDENT	5000
7844	TURNER	SALESMAN	1500
7902	FORD	ANALYST	3000

Pattern Matching Search …

- SQL has two special pattern matching symbols:
  - **%**: sequence of zero or more characters;
  - _ (underscore): any single character.

- **LIKE** '%dd%' means a sequence of characters of any length containing '*dd*'.

Pattern Matching Search …

Selecting all the records whose column value is a member of the set specified in the WHERE clause.

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	17-DEC-80	800		20
7499	ALLEN	SALESMAN	7698	20-FEB-81	1600	300	30
7521	WARD	SALESMAN	7698	22-FEB-81	1250	500	30
7566	JONES	MANAGER	7839	02-APR-81	2975		20
7654	MARTIN	SALESMAN	7698	28-SEP-81	1250	1400	30
7698	BLAKE	MANAGER	7839	01-MAY-81	2850		30
7782	CLARK	MANAGER	7839	09-JUN-81	2450		10
7788	SCOTT	ANALYST	7566	19-APR-87	3000		20
7839	KING	PRESIDENT		17-NOV-81	5000		10
7844	TURNER	SALESMAN	7698	08-SEP-81	1500	0	30
7876	ADAMS	CLERK	7788	23-MAY-87	1100		20
7900	JAMES	CLERK	7698	03-DEC-81	950		30
7902	FORD	ANALYST	7566	03-DEC-81	3000		
7934	MILLER	CLERK	7782	23-JAN-82	1300		10

```
SELECT empno, ename, job, sal
FROM   emp
WHERE job LIKE 'A%';
```

EMPNO	ENAME	JOB	SAL
7788	SCOTT	ANALYST	3000
7902	FORD	ANALYST	3000

```
SELECT empno, ename, job, sal
FROM   emp
WHERE job LIKE '%AN%';
```

EMPNO	ENAME	JOB	SAL
7499	ALLEN	SALESMAN	1600
7521	WARD	SALESMAN	1250
7566	JONES	MANAGER	2975
7654	MARTIN	SALESMAN	1250
7698	BLAKE	MANAGER	2850
7782	CLARK	MANAGER	2450
7788	SCOTT	ANALYST	3000
7844	TURNER	SALESMAN	1500
7902	FORD	ANALYST	3000

NULL SEARCH

Selecting all employees with Commission and without Commission. Means Commission is available or not available

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	17-DEC-80	800		20
7499	ALLEN	SALESMAN	7698	20-FEB-81	1600	300	30
7521	WARD	SALESMAN	7698	22-FEB-81	1250	500	30
7566	JONES	MANAGER	7839	02-APR-81	2975		20
7654	MARTIN	SALESMAN	7698	28-SEP-81	1250	1400	30
7698	BLAKE	MANAGER	7839	01-MAY-81	2850		30
7782	CLARK	MANAGER	7839	09-JUN-81	2450		10
7788	SCOTT	ANALYST	7566	19-APR-87	3000		20
7839	KING	PRESIDENT		17-NOV-81	5000		10
7844	TURNER	SALESMAN	7698	08-SEP-81	1500	0	30
7876	ADAMS	CLERK	7788	23-MAY-87	1100		20
7900	JAMES	CLERK	7698	03-DEC-81	950		30
7902	FORD	ANALYST	7566	03-DEC-81	3000		
7934	MILLER	CLERK	7782	23-JAN-82	1300		10

```
SELECT empno, ename, sal, comm
FROM   emp
WHERE comm is NOT NULL;
```

EMPNO	ENAME	SAL	COMM
7499	ALLEN	1600	300
7521	WARD	1250	500
7654	MARTIN	1250	1400
7844	TURNER	1500	0

```
SELECT empno, ename, sal, comm
FROM   emp
WHERE comm is NULL;
```

REMOVING DUPLICATE ROWS

Selecting all employees with Commission and without Commission. Means Commission is available or not available

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	17-DEC-80	800		20
7499	ALLEN	SALESMAN	7698	20-FEB-81	1600	300	30
7521	WARD	SALESMAN	7698	22-FEB-81	1250	500	30
7566	JONES	MANAGER	7839	02-APR-81	2975		20
7654	MARTIN	SALESMAN	7698	28-SEP-81	1250	1400	30
7698	BLAKE	MANAGER	7839	01-MAY-81	2850		30
7782	CLARK	MANAGER	7839	09-JUN-81	2450		10
7788	SCOTT	ANALYST	7566	19-APR-87	3000		20
7839	KING	PRESIDENT		17-NOV-81	5000		10
7844	TURNER	SALESMAN	7698	08-SEP-81	1500	0	30
7876	ADAMS	CLERK	7788	23-MAY-87	1100		20
7900	JAMES	CLERK	7698	03-DEC-81	950		30
7902	FORD	ANALYST	7566	03-DEC-81	3000		
7934	MILLER	CLERK	7782	23-JAN-82	1300		10

```
SELECT job
FROM   emp;
```

```
JOB
---------
CLERK
SALESMAN
SALESMAN
MANAGER
SALESMAN
MANAGER
MANAGER
ANALYST
PRESIDENT
SALESMAN
CLERK
CLERK
ANALYST
CLERK
```

```
SELECT DISTINCT job
FROM   emp;
```

```
JOB
---------
CLERK
SALESMAN
PRESIDENT
MANAGER
ANALYST
```

Sorting

- The ORDER BY clause specifies an order for displaying the result of a query.
  - SQL allows the user to order the tuples in the result of a query by the values of one or more attributes; the default order is ascending or increasing.
  - The keyword **DESC** is specified to sort in a descending order of values while the keyword **ASC** can be used to specify ascending order explicitly.
  - The sorting will be applied alphabetically or numerically depending on the type of the column attribute.

SORTING

Sorting records in Ascending, Descending orders

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	17-DEC-80	800		20
7499	ALLEN	SALESMAN	7698	20-FEB-81	1600	300	30
7521	WARD	SALESMAN	7698	22-FEB-81	1250	500	30
7566	JONES	MANAGER	7839	02-APR-81	2975		20
7654	MARTIN	SALESMAN	7698	28-SEP-81	1250	1400	30
7698	BLAKE	MANAGER	7839	01-MAY-81	2850		30
7782	CLARK	MANAGER	7839	09-JUN-81	2450		10
7788	SCOTT	ANALYST	7566	19-APR-87	3000		20
7839	KING	PRESIDENT		17-NOV-81	5000		10
7844	TURNER	SALESMAN	7698	08-SEP-81	1500	0	30
7876	ADAMS	CLERK	7788	23-MAY-87	1100		20
7900	JAMES	CLERK	7698	03-DEC-81	950		30
7902	FORD	ANALYST	7566	03-DEC-81	3000		
7934	MILLER	CLERK	7782	23-JAN-82	1300		10

```
SELECT empno, ename, job, comm
FROM   emp
ORDER BY ename DESC, comm ASC;
```

EMPNO	ENAME	JOB	COMM
7521	WARD	SALESMAN	500
7844	TURNER	SALESMAN	0
7369	SMITH	CLERK	
7788	SCOTT	ANALYST	
7934	MILLER	CLERK	
7654	MARTIN	SALESMAN	1400
7839	KING	PRESIDENT	
7566	JONES	MANAGER	
7900	JAMES	CLERK	
7902	FORD	ANALYST	
7782	CLARK	MANAGER	
7698	BLAKE	MANAGER	
7499	ALLEN	SALESMAN	300
7876	ADAMS	CLERK	

```
SELECT empno, ename, sal, comm
FROM   emp
ORDER BY job, sal DESC;
```

Aggregation …

- ISO standard defines five aggregate functions:
  - **COUNT** returns number of values in a specified column.
  - **SUM** returns sum of values in a specified column.
  - **AVG** returns average of values in a specified column.
  - **MIN** returns smallest value in a specified column.
  - **MAX** returns largest value in a specified column.

Aggregation …

- Each operates on a single column of a table and return single value.
- COUNT, MIN, and MAX apply to numeric and non-numeric fields, but SUM and AVG may be used on numeric fields only.
- Apart from COUNT(*), each function eliminates nulls first and operates only on remaining non-null values.
- COUNT(*) counts all rows of a table, regardless of whether nulls or duplicate values occur.
- Can use DISTINCT before column name to eliminate duplicates.

Aggregation …

- DISTINCT has no effect with MIN/MAX, but may have with SUM/AVG.
- Aggregate functions can be used only in SELECT list and in HAVING clause.
- If SELECT list includes an aggregate function and there is no GROUP BY clause, then SELECT list cannot reference a column with an aggregate function. For example, following is illegal:

```
SELECT job, COUNT(*)
  FROM emp;
```

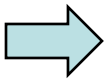
Example: COUNT

How many rows in table EMP?

How many job titles in EMP?

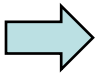
EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	17-DEC-80	800		20
7499	ALLEN	SALESMAN	7698	20-FEB-81	1600	300	30
7521	WARD	SALESMAN	7698	22-FEB-81	1250	500	30
7566	JONES	MANAGER	7839	02-APR-81	2975		20
7654	MARTIN	SALESMAN	7698	28-SEP-81	1250	1400	30
7698	BLAKE	MANAGER	7839	01-MAY-81	2850		30
7782	CLARK	MANAGER	7839	09-JUN-81	2450		10
7788	SCOTT	ANALYST	7566	19-APR-87	3000		20
7839	KING	PRESIDENT		17-NOV-81	5000		10
7844	TURNER	SALESMAN	7698	08-SEP-81	1500	0	30
7876	ADAMS	CLERK	7788	23-MAY-87	1100		20
7900	JAMES	CLERK	7698	03-DEC-81	950		30
7902	FORD	ANALYST	7566	03-DEC-81	3000		
7934	MILLER	CLERK	7782	23-JAN-82	1300		10

```
SELECT COUNT(*)
FROM   emp;
```



COUNT(*)
14

```
SELECT COUNT(DISTINCT job)
FROM   emp;
```



COUNT(DISTINCTJOB)
5

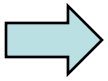
Example: SUM

Find the total of all salaries in EMP?

Find the total of salaries of employees of dept 10?

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	17-DEC-80	800		20
7499	ALLEN	SALESMAN	7698	20-FEB-81	1600	300	30
7521	WARD	SALESMAN	7698	22-FEB-81	1250	500	30
7566	JONES	MANAGER	7839	02-APR-81	2975		20
7654	MARTIN	SALESMAN	7698	28-SEP-81	1250	1400	30
7698	BLAKE	MANAGER	7839	01-MAY-81	2850		30
7782	CLARK	MANAGER	7839	09-JUN-81	2450		10
7788	SCOTT	ANALYST	7566	19-APR-87	3000		20
7839	KING	PRESIDENT		17-NOV-81	5000		10
7844	TURNER	SALESMAN	7698	08-SEP-81	1500	0	30
7876	ADAMS	CLERK	7788	23-MAY-87	1100		20
7900	JAMES	CLERK	7698	03-DEC-81	950		30
7902	FORD	ANALYST	7566	03-DEC-81	3000		
7934	MILLER	CLERK	7782	23-JAN-82	1300		10

```
SELECT SUM(sal)
FROM   emp;
```



SUM(SAL)
29025

```
SELECT SUM(sal) TotSalary
FROM   emp
WHERE deptno=10;
```

TOTSALARY
8750

Example: MIN, MAX, AVG

Find the minimum, maximum, and average salary of all employees

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	17-DEC-80	800		20
7499	ALLEN	SALESMAN	7698	20-FEB-81	1600	300	30
7521	WARD	SALESMAN	7698	22-FEB-81	1250	500	30
7566	JONES	MANAGER	7839	02-APR-81	2975		20
7654	MARTIN	SALESMAN	7698	28-SEP-81	1250	1400	30
7698	BLAKE	MANAGER	7839	01-MAY-81	2850		30
7782	CLARK	MANAGER	7839	09-JUN-81	2450		10
7788	SCOTT	ANALYST	7566	19-APR-87	3000		20
7839	KING	PRESIDENT		17-NOV-81	5000		10
7844	TURNER	SALESMAN	7698	08-SEP-81	1500	0	30
7876	ADAMS	CLERK	7788	23-MAY-87	1100		20
7900	JAMES	CLERK	7698	03-DEC-81	950		30
7902	FORD	ANALYST	7566	03-DEC-81	3000		
7934	MILLER	CLERK	7782	23-JAN-82	1300		10

```
SELECT MIN(sal) Min_sal,
       MAX(sal) Max_sal,
       AVG(sal) Average_sal
FROM emp;
```

MIN_SAL	MAX_SAL	AVERAGE_SAL
800	5000	2073.21429

Grouping

- Use **GROUP BY** clause to get sub-totals.
- SELECT and GROUP BY closely integrated: each item in SELECT list must be **single-valued per group**, and SELECT clause may only contain:
  - Column names.
  - Aggregate functions.
  - Constants.
  - An expression involving combinations of the above.
- All column names in SELECT list must appear in GROUP BY clause unless name is used only in an aggregate function.
- If WHERE is used with GROUP BY, WHERE is applied first, then groups are formed from remaining rows satisfying predicate.
- ISO considers two nulls to be equal for purposes of GROUP BY.

Example: GROUPING

Find the number of employees in each department.

Find the number of employees in each department & with job title.

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	17-DEC-80	800		20
7499	ALLEN	SALESMAN	7698	20-FEB-81	1600	300	30
7521	WARD	SALESMAN	7698	22-FEB-81	1250	500	30
7566	JONES	MANAGER	7839	02-APR-81	2975		20
7654	MARTIN	SALESMAN	7698	28-SEP-81	1250	1400	30
7698	BLAKE	MANAGER	7839	01-MAY-81	2850		30
7782	CLARK	MANAGER	7839	09-JUN-81	2450		10
7788	SCOTT	ANALYST	7566	19-APR-87	3000		20
7839	KING	PRESIDENT		17-NOV-81	5000		10
7844	TURNER	SALESMAN	7698	08-SEP-81	1500	0	30
7876	ADAMS	CLERK	7788	23-MAY-87	1100		20
7900	JAMES	CLERK	7698	03-DEC-81	950		30
7902	FORD	ANALYST	7566	03-DEC-81	3000		
7934	MILLER	CLERK	7782	23-JAN-82	1300		10

Aggregate & non-aggregate columns

```
SELECT deptno, COUNT(*)
FROM emp
GROUP BY deptno;
```

DEPTNO	COUNT(*)
30	6
	1
20	4
10	3

```
SELECT deptno, job, COUNT(*)
FROM emp
WHERE deptno IN (10,20)
GROUP BY deptno, job;
```

DEPTNO	JOB	COUNT(*)
20	CLERK	2
20	MANAGER	1
10	PRESIDENT	1
10	CLERK	1
10	MANAGER	1
20	ANALYST	1

Example: GROUPING

Executable SQL with Group by but meaningless output.

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	17-DEC-80	800		20
7499	ALLEN	SALESMAN	7698	20-FEB-81	1600	300	30
7521	WARD	SALESMAN	7698	22-FEB-81	1250	500	30
7566	JONES	MANAGER	7839	02-APR-81	2975		20
7654	MARTIN	SALESMAN	7698	28-SEP-81	1250	1400	30
7698	BLAKE	MANAGER	7839	01-MAY-81	2850		30
7782	CLARK	MANAGER	7839	09-JUN-81	2450		10
7788	SCOTT	ANALYST	7566	19-APR-87	3000		20
7839	KING	PRESIDENT		17-NOV-81	5000		10
7844	TURNER	SALESMAN	7698	08-SEP-81	1500	0	30
7876	ADAMS	CLERK	7788	23-MAY-87	1100		20
7900	JAMES	CLERK	7698	03-DEC-81	950		30
7902	FORD	ANALYST	7566	03-DEC-81	3000		
7934	MILLER	CLERK	7782	23-JAN-82	1300		10

```
select max(sal)
from emp
group by job;
```

```
  MAX(SAL)
----------
      1300
      1600
      5000
      2975
      3000
```

Restricting Groups

- **HAVING clause** is designed for use with GROUP BY clause to restrict groups that appear in final result table.

- Similar to WHERE, but WHERE filters individual rows whereas HAVING filters groups.

- Column names in HAVING clause must also appear in the GROUP BY list or be contained within an aggregate function.

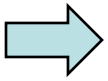
Example: Restricting Groups

Group By with conditions

You can use WHERE before Group By

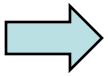
EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	17-DEC-80	800		20
7499	ALLEN	SALESMAN	7698	20-FEB-81	1600	300	30
7521	WARD	SALESMAN	7698	22-FEB-81	1250	500	30
7566	JONES	MANAGER	7839	02-APR-81	2975		20
7654	MARTIN	SALESMAN	7698	28-SEP-81	1250	1400	30
7698	BLAKE	MANAGER	7839	01-MAY-81	2850		30
7782	CLARK	MANAGER	7839	09-JUN-81	2450		10
7788	SCOTT	ANALYST	7566	19-APR-87	3000		20
7839	KING	PRESIDENT		17-NOV-81	5000		10
7844	TURNER	SALESMAN	7698	08-SEP-81	1500	0	30
7876	ADAMS	CLERK	7788	23-MAY-87	1100		20
7900	JAMES	CLERK	7698	03-DEC-81	950		30
7902	FORD	ANALYST	7566	03-DEC-81	3000		
7934	MILLER	CLERK	7782	23-JAN-82	1300		10

```
SELECT job, SUM(sal) SUM_SAL
FROM   emp
GROUP BY job;
```



JOB	SUM_SAL
CLERK	4150
SALESMAN	5600
PRESIDENT	5000
MANAGER	8275
ANALYST	6000

```
SELECT job, SUM(sal) SUM_SAL
FROM   emp
GROUP BY job
HAVING SUM(sal)>5000;
```



JOB	SUM_SAL
SALESMAN	5600
MANAGER	8275
ANALYST	6000

Aliasing Table Names

- A table alias is created by directly placing an alias after the table name in the FROM clause.
- The advantage of using a table alias when performing JOIN is readily apparent when we discuss JOIN later.

- For example in the following example we will refer to departments table as **d** and employee table as **e**.

```
SELECT d.dname
FROM   dept d
WHERE d.deptno = 10;
```

```
SELECT e.ename, e.sal
FROM   emp e
WHERE e.deptno = 10;
```

Nested or Sub Queries

- Some SQL statements can have a SELECT embedded within them.
- A subselect can be used in WHERE and HAVING clauses of an outer SELECT, where it is called a ***nested query*** or a ***subquery***.
- Subselects may also appear in INSERT, UPDATE, and DELETEs.

Nested or Sub Queries

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON
50	MAINTENANCE	DALLAS

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	17-DEC-80	800		20
7499	ALLEN	SALESMAN	7698	20-FEB-81	1600	300	30
7521	WARD	SALESMAN	7698	22-FEB-81	1250	500	30
7566	JONES	MANAGER	7839	02-APR-81	2975		20
7654	MARTIN	SALESMAN	7698	28-SEP-81	1250	1400	30
7698	BLAKE	MANAGER	7839	01-MAY-81	2850		30
7782	CLARK	MANAGER	7839	09-JUN-81	2450		10
7788	SCOTT	ANALYST	7566	19-APR-87	3000		20
7839	KING	PRESIDENT		17-NOV-81	5000		10
7844	TURNER	SALESMAN	7698	08-SEP-81	1500	0	30
7876	ADAMS	CLERK	7788	23-MAY-87	1100		20
7900	JAMES	CLERK	7698	03-DEC-81	950		30
7902	FORD	ANALYST	7566	03-DEC-81	3000		
7934	MILLER	CLERK	7782	23-JAN-82	1300		10

List of employees who are belong to department name Research

```
SELECT *
FROM   emp
WHERE deptno IN (
                    SELECT deptno
                    FROM dept
                     WHERE dname = 'RESEARCH');
```

Inner select Executed First

Outer select

Nested or Sub Queries

List of employees whose salary is more than the average salary of department 30

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	17-DEC-80	800		20
7499	ALLEN	SALESMAN	7698	20-FEB-81	1600	300	30
7521	WARD	SALESMAN	7698	22-FEB-81	1250	500	30
7566	JONES	MANAGER	7839	02-APR-81	2975		20
7654	MARTIN	SALESMAN	7698	28-SEP-81	1250	1400	30
7698	BLAKE	MANAGER	7839	01-MAY-81	2850		30
7782	CLARK	MANAGER	7839	09-JUN-81	2450		10
7788	SCOTT	ANALYST	7566	19-APR-87	3000		20
7839	KING	PRESIDENT		17-NOV-81	5000		10
7844	TURNER	SALESMAN	7698	08-SEP-81	1500	0	30
7876	ADAMS	CLERK	7788	23-MAY-87	1100		20
7900	JAMES	CLERK	7698	03-DEC-81	950		30
7902	FORD	ANALYST	7566	03-DEC-81	3000		
7934	MILLER	CLERK	7782	23-JAN-82	1300		10

```
SELECT *
FROM   emp
WHERE sal > (
                    SELECT AVG(sal)
                    FROM emp
                     WHERE deptno=30
                );
```

Single Record Executed First

Nested Query Rules

- ORDER BY clause may not be used in a subquery (although it may be used in outermost SELECT).
- Subquery SELECT list must consist of a single column name or expression, except for subqueries that use EXISTS.
- By default, column names refer to table name in FROM clause of subquery. Can refer to a table in FROM using an *alias*.
- When subquery is an operand in a comparison, subquery must appear on right-hand side.
- A subquery may not be used as an operand in an expression.

Example: Nested or Sub Queries

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON
50	MAINTENANCE	DALLAS

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	17-DEC-80	800		20
7499	ALLEN	SALESMAN	7698	20-FEB-81	1600	300	30
7521	WARD	SALESMAN	7698	22-FEB-81	1250	500	30
7566	JONES	MANAGER	7839	02-APR-81	2975		20
7654	MARTIN	SALESMAN	7698	28-SEP-81	1250	1400	30
7698	BLAKE	MANAGER	7839	01-MAY-81	2850		30
7782	CLARK	MANAGER	7839	09-JUN-81	2450		10
7788	SCOTT	ANALYST	7566	19-APR-87	3000		20
7839	KING	PRESIDENT		17-NOV-81	5000		10
7844	TURNER	SALESMAN	7698	08-SEP-81	1500	0	30
7876	ADAMS	CLERK	7788	23-MAY-87	1100		20
7900	JAMES	CLERK	7698	03-DEC-81	950		30
7902	FORD	ANALYST	7566	03-DEC-81	3000		
7934	MILLER	CLERK	7782	23-JAN-82	1300		10

List of employees whose salaries is higher than the salary of at least one employee from department Research

```
SELECT *
FROM   emp
WHERE sal > (  SELECT MIN(sal)
                 FROM emp
                 WHERE deptno = ( SELECT deptno
                                  FROM dept
                                  WHERE dname = 'RESEARCH‘));
```

Example: Nested or Sub Queries

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON
50	MAINTENANCE	DALLAS

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	17-DEC-80	800		20
7499	ALLEN	SALESMAN	7698	20-FEB-81	1600	300	30
7521	WARD	SALESMAN	7698	22-FEB-81	1250	500	30
7566	JONES	MANAGER	7839	02-APR-81	2975		20
7654	MARTIN	SALESMAN	7698	28-SEP-81	1250	1400	30
7698	BLAKE	MANAGER	7839	01-MAY-81	2850		30
7782	CLARK	MANAGER	7839	09-JUN-81	2450		10
7788	SCOTT	ANALYST	7566	19-APR-87	3000		20
7839	KING	PRESIDENT		17-NOV-81	5000		10
7844	TURNER	SALESMAN	7698	08-SEP-81	1500	0	30
7876	ADAMS	CLERK	7788	23-MAY-87	1100		20
7900	JAMES	CLERK	7698	03-DEC-81	950		30
7902	FORD	ANALYST	7566	03-DEC-81	3000		
7934	MILLER	CLERK	7782	23-JAN-82	1300		10

List of employees whose salaries is higher than the salary of every employee from department Research

```
SELECT *
FROM   emp
WHERE sal > (  SELECT MAX(sal)
                  FROM emp
                  WHERE deptno = ( SELECT deptno
                                          FROM dept
                                          WHERE dname = 'RESEARCH‘));
```

Join

- Can use subqueries provided result columns come from same table.
- If result columns come from more than one table must use a join.
- To perform join, include more than one table in FROM clause.
- Use comma as separator and typically include WHERE clause to specify join column(s).

- Also possible to use an alias for a table named in FROM clause.
- Alias is separated from table name with a space.
- Alias can be used to qualify column names when there is ambiguity.

Example: Join (Inner Join)

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON
50	MAINTENANCE	DALLAS

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	17-DEC-80	800		20
7499	ALLEN	SALESMAN	7698	20-FEB-81	1600	300	30
7521	WARD	SALESMAN	7698	22-FEB-81	1250	500	30
7566	JONES	MANAGER	7839	02-APR-81	2975		20
7654	MARTIN	SALESMAN	7698	28-SEP-81	1250	1400	30
7698	BLAKE	MANAGER	7839	01-MAY-81	2850		30
7782	CLARK	MANAGER	7839	09-JUN-81	2450		10
7788	SCOTT	ANALYST	7566	19-APR-87	3000		20
7839	KING	PRESIDENT		17-NOV-81	5000		10
7844	TURNER	SALESMAN	7698	08-SEP-81	1500	0	30
7876	ADAMS	CLERK	7788	23-MAY-87	1100		20
7900	JAMES	CLERK	7698	03-DEC-81	950		30
7902	FORD	ANALYST	7566	03-DEC-81	3000		
7934	MILLER	CLERK	7782	23-JAN-82	1300		10

The default type of join is inner join, where arrow is included in the result only if matching row exists in the other relation.

```
SELECT e.empno, e.deptno, d.dname, d.deptno
FROM   dept d, emp e
WHERE d.deptno=e.deptno
AND d.dname IN ('SALES','RESEARCH');
```

EMPNO	DEPTNO	DNAME	DEPTNO
7566	20	RESEARCH	20
7788	20	RESEARCH	20
7369	20	RESEARCH	20
7876	20	RESEARCH	20
7499	30	SALES	30
7900	30	SALES	30
7844	30	SALES	30
7521	30	SALES	30
7698	30	SALES	30
7654	30	SALES	30

PK-FK never makes default join, must specify Join in SQL

Example: Join (Inner Join) …

- To obtain correct rows, include only those rows from both tables that have identical values in the dno columns: a.dno = b.dno.

- These two columns are the matching columns for two tables.

- This type of join is also called **inner join** and they equivalent to equi-join in relational algebra.

Computing a Join

- Procedure for generating results of a SELECT with a join are:
  1. Form Cartesian product of the tables named in FROM clause.
  2. If there is a WHERE clause, apply the search condition to each row of the product table, retaining those rows that satisfy the condition.
  3. For each remaining row, determine the value of each item in the SELECT list to produce a single row in the result table.
  4. If SELECT DISTINCT has been specified, eliminate any duplicate rows from the result table.
  5. If there is an ORDER BY clause, sort the result table as required.

Outer Joins

- With an inner join, if one row of a table is unmatched, row is omitted from result table.
- The outer join operations retain rows that do not satisfy the join condition.
- There are three types of OUTER JOIN
  - Left Outer Join
  - Right Outer Join
  - Full Outer Join
- Lets discuss inner join then we will come back to outer join.
- Query execution (Performance) is much better than other joins

For Outer Join

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON
50	MAINTENANCE	DALLAS

Inner join of departments and lecturers tables will result in the following output.

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	17-DEC-80	800		20
7499	ALLEN	SALESMAN	7698	20-FEB-81	1600	300	30
7521	WARD	SALESMAN	7698	22-FEB-81	1250	500	30
7566	JONES	MANAGER	7839	02-APR-81	2975		20
7654	MARTIN	SALESMAN	7698	28-SEP-81	1250	1400	30
7698	BLAKE	MANAGER	7839	01-MAY-81	2850		30
7782	CLARK	MANAGER	7839	09-JUN-81	2450		10
7788	SCOTT	ANALYST	7566	19-APR-87	3000		20
7839	KING	PRESIDENT		17-NOV-81	5000		10
7844	TURNER	SALESMAN	7698	08-SEP-81	1500	0	30
7876	ADAMS	CLERK	7788	23-MAY-87	1100		20
7900	JAMES	CLERK	7698	03-DEC-81	950		30
7902	FORD	ANALYST	7566	03-DEC-81	3000		
7934	MILLER	CLERK	7782	23-JAN-82	1300		10

```
SELECT e.empno, e.ename, e.sal,
e.deptno, d.dname, d.deptno
FROM   emp e, dept d
WHERE e.deptno=d.deptno;
```

EMPNO	ENAME	SAL	DEPTNO	DNAME	DEPTNO
7782	CLARK	2450	10	ACCOUNTING	10
7839	KING	5000	10	ACCOUNTING	10
7934	MILLER	1300	10	ACCOUNTING	10
7566	JONES	2975	20	RESEARCH	20
7788	SCOTT	3000	20	RESEARCH	20
7369	SMITH	800	20	RESEARCH	20
7876	ADAMS	1100	20	RESEARCH	20
7499	ALLEN	1600	30	SALES	30
7900	JAMES	950	30	SALES	30
7844	TURNER	1500	30	SALES	30
7521	WARD	1250	30	SALES	30
7698	BLAKE	2850	30	SALES	30
7654	MARTIN	1250	30	SALES	30

Outer Join …

- Result table has two rows where the deptno are the same.
- There are no rows corresponding to Maintenance or Ford.
- To include unmatched rows in result table, use an outer joins.
- Benefit: Fastest execution of query

Example: Left Outer Join

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON
50	MAINTENANCE	DALLAS

List of employees who are belong to departments and who are not belong to any department

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	17-DEC-80	800		20
7499	ALLEN	SALESMAN	7698	20-FEB-81	1600	300	30
7521	WARD	SALESMAN	7698	22-FEB-81	1250	500	30
7566	JONES	MANAGER	7839	02-APR-81	2975		20
7654	MARTIN	SALESMAN	7698	28-SEP-81	1250	1400	30
7698	BLAKE	MANAGER	7839	01-MAY-81	2850		30
7782	CLARK	MANAGER	7839	09-JUN-81	2450		10
7788	SCOTT	ANALYST	7566	19-APR-87	3000		20
7839	KING	PRESIDENT		17-NOV-81	5000		10
7844	TURNER	SALESMAN	7698	08-SEP-81	1500	0	30
7876	ADAMS	CLERK	7788	23-MAY-87	1100		20
7900	JAMES	CLERK	7698	03-DEC-81	950		30
7902	FORD	ANALYST	7566	03-DEC-81	3000		
7934	MILLER	CLERK	7782	23-JAN-82	1300		10

```
SELECT e.empno, e.ename, e.sal,
e.deptno, d.dname, d.deptno
FROM   emp e, dept d
WHERE e.deptno=d.deptno (+);
```

EMPNO	ENAME	SAL	DEPTNO	DNAME	DEPTNO
7782	CLARK	2450	10	ACCOUNTING	10
7839	KING	5000	10	ACCOUNTING	10
7934	MILLER	1300	10	ACCOUNTING	10
7566	JONES	2975	20	RESEARCH	20
7788	SCOTT	3000	20	RESEARCH	20
7369	SMITH	800	20	RESEARCH	20
7876	ADAMS	1100	20	RESEARCH	20
7499	ALLEN	1600	30	SALES	30
7900	JAMES	950	30	SALES	30
7844	TURNER	1500	30	SALES	30
7521	WARD	1250	30	SALES	30
7698	BLAKE	2850	30	SALES	30
7654	MARTIN	1250	30	SALES	30
7902	FORD	3000			

Example: Right Outer Join

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON
50	MAINTENANCE	DALLAS

List of departments which are departments of employees and which are not.

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	17-DEC-80	800		20
7499	ALLEN	SALESMAN	7698	20-FEB-81	1600	300	30
7521	WARD	SALESMAN	7698	22-FEB-81	1250	500	30
7566	JONES	MANAGER	7839	02-APR-81	2975		20
7654	MARTIN	SALESMAN	7698	28-SEP-81	1250	1400	30
7698	BLAKE	MANAGER	7839	01-MAY-81	2850		30
7782	CLARK	MANAGER	7839	09-JUN-81	2450		10
7788	SCOTT	ANALYST	7566	19-APR-87	3000		20
7839	KING	PRESIDENT		17-NOV-81	5000		10
7844	TURNER	SALESMAN	7698	08-SEP-81	1500	0	30
7876	ADAMS	CLERK	7788	23-MAY-87	1100		20
7900	JAMES	CLERK	7698	03-DEC-81	950		30
7902	FORD	ANALYST	7566	03-DEC-81	3000		
7934	MILLER	CLERK	7782	23-JAN-82	1300		10

```
SELECT e.empno, e.ename, e.sal,
e.deptno, d.dname, d.deptno
FROM   emp e, dept d
WHERE e.deptno (+)=d.deptno;
```

EMPNO	ENAME	SAL	DEPTNO	DNAME	DEPTNO
7782	CLARK	2450	10	ACCOUNTING	10
7839	KING	5000	10	ACCOUNTING	10
7934	MILLER	1300	10	ACCOUNTING	10
7566	JONES	2975	20	RESEARCH	20
7788	SCOTT	3000	20	RESEARCH	20
7369	SMITH	800	20	RESEARCH	20
7876	ADAMS	1100	20	RESEARCH	20
7499	ALLEN	1600	30	SALES	30
7900	JAMES	950	30	SALES	30
7844	TURNER	1500	30	SALES	30
7521	WARD	1250	30	SALES	30
7698	BLAKE	2850	30	SALES	30
7654	MARTIN	1250	30	SALES	30
				OPERATIONS	40
				MAINTENANCE	50

Example: Full Outer Join

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON
50	MAINTENANCE	DALLAS

List of employees and department which are matched or unmatched from both tables

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	17-DEC-80	800		20
7499	ALLEN	SALESMAN	7698	20-FEB-81	1600	300	30
7521	WARD	SALESMAN	7698	22-FEB-81	1250	500	30
7566	JONES	MANAGER	7839	02-APR-81	2975		20
7654	MARTIN	SALESMAN	7698	28-SEP-81	1250	1400	30
7698	BLAKE	MANAGER	7839	01-MAY-81	2850		30
7782	CLARK	MANAGER	7839	09-JUN-81	2450		10
7788	SCOTT	ANALYST	7566	19-APR-87	3000		20
7839	KING	PRESIDENT		17-NOV-81	5000		10
7844	TURNER	SALESMAN	7698	08-SEP-81	1500	0	30
7876	ADAMS	CLERK	7788	23-MAY-87	1100		20
7900	JAMES	CLERK	7698	03-DEC-81	950		30
7902	FORD	ANALYST	7566	03-DEC-81	3000		
7934	MILLER	CLERK	7782	23-JAN-82	1300		10

```
(SELECT e.empno, e.ename, e.sal,
e.deptno, d.dname, d.deptno
FROM   emp e, dept d
WHERE e.deptno=d.deptno (+) )
UNION
(SELECT e.empno, e.ename, e.sal,
e.deptno, d.dname, d.deptno
FROM   emp e, dept d
WHERE e.deptno (+) =d.deptno);
```

EMPNO	ENAME	SAL	DEPTNO	DNAME	DEPTNO
7782	CLARK	2450	10	ACCOUNTING	10
7839	KING	5000	10	ACCOUNTING	10
7934	MILLER	1300	10	ACCOUNTING	10
7566	JONES	2975	20	RESEARCH	20
7788	SCOTT	3000	20	RESEARCH	20
7369	SMITH	800	20	RESEARCH	20
7876	ADAMS	1100	20	RESEARCH	20
7499	ALLEN	1600	30	SALES	30
7900	JAMES	950	30	SALES	30
7844	TURNER	1500	30	SALES	30
7521	WARD	1250	30	SALES	30
7698	BLAKE	2850	30	SALES	30
7654	MARTIN	1250	30	SALES	30
7902	FORD	3000			
				OPERATIONS	40
				MAINTENANCE	50

Example: Left Outer Join (3-Tables)…

```
SQL>
SQL> select d.dname, e.empno, e.ename, p.projno, p.noofhrs
  2  from dept d, emp e, projassign p
  3  where d.deptno=e.deptno(+)
  4  and e.empno=p.empno(+);

DNAME          EMPNO ENAME      PROJNO    NOOFHRS
-------------- ----------  ---------- ------ ----------
RESEARCH             7369 SMITH      PM2210         20
RESEARCH             7369 SMITH      PM2211          5
SALES                7654 MARTIN     PM2278         15
RESEARCH             7788 SCOTT      PM2278         25
RESEARCH             7788 SCOTT      PM2211         12
OPERATIONS
SALES                7844 TURNER
ACCOUNTING           7839 KING
ACCOUNTING           7782 CLARK
SALES                7521 WARD
SALES                7698 BLAKE
RESEARCH             7566 JONES
SALES                7499 ALLEN
ACCOUNTING           7934 MILLER
RESEARCH             7876 ADAMS
SALES                7900 JAMES

16 rows selected.

SQL>
```

Characteristic of Outer Join

- Left Outer Join:
  - Includes those rows of first (left) table unmatched with rows from second (right) table.
  - Columns from second table are filled with NULLs.

- Right outer Join :
  - includes those rows of second (right) table that are unmatched with rows from first (left) table.
  - Columns from first table are filled with NULLs.

- Full Outer Join:
  - Is the UNION of both left and right outer joins.

Examples: Left, Right and Full Outer Joins

```
select d.dname, e.ename, e.sal
from dept d, emp e
where d.deptno = e.deptno (+)
order by d.dname;

select d.dname, e.ename, e.sal
from dept d LEFT OUTER JOIN emp e
ON d.deptno = e.deptno
order by d.dname;
-------------------------------------
select d.dname, e.ename, e.sal
from dept d, emp e
where d.deptno (+) = e.deptno
order by d.dname;

select d.dname, e.ename, e.sal
from dept d RIGHT OUTER JOIN emp e
ON d.deptno = e.deptno
order by d.dname;
-------------------------------------
select d.dname, e.ename, e.sal
from dept d FULL OUTER JOIN emp e
ON d.deptno = e.deptno
order by d.dname;
```

Union, Intersect, and Difference

- Can use normal set operations of union, intersection, and difference to combine results of two or more queries into a single result table.
- Union of two tables, A and B, is table containing all rows in either A or B or both.
- Intersection is table containing all rows common to both A and B.
- Difference is table containing all rows in A but not in B.
- Two tables must be *union compatible*.
- If ALL specified, result can include duplicate rows

Input Tables

Example: Use of UNION

- **List all the department nos 10, 20 salaries. Remove duplicates**

```
SELECT sal
FROM emp
WHERE deptno = 10
UNION
SELECT sal
FROM emp
WHERE deptno = 20;
```

- **List all the department nos 10, 20 salaries. Including duplicates**

```
SELECT sal
FROM emp
WHERE deptno = 10
UNION ALL
SELECT sal
FROM emp
WHERE deptno = 20;
```

Example: Use of UNION …

- **List all the ICS and COE faculty salaries. Remove duplicates**

```
SELECT salary
FROM lecturers
WHERE dno =
( SELECT dno
  FROM departments
  WHERE dname= 'ICS'
)
UNION
SELECT salary
FROM lecturers
WHERE dno =
( SELECT dno
  FROM departments
  WHERE dname= 'COE'
)
```

- **List all the ICS and COE faculty salaries. Include duplicates**

```
SELECT salary
FROM lecturers
WHERE dno =
( SELECT dno
  FROM departments
  WHERE dname= 'ICS'
)
UNION ALL
SELECT salary
FROM lecturers
WHERE dno =
( SELECT dno
  FROM departments
  WHERE dname= 'COE'
)
```

Input Tables

Example: Use of DIFFERENCE

- **List salaries that are taken by ICS and not COE lecturers.**

```
SELECT salary
FROM lecturers
WHERE dno = (
                SELECT dno
                FROM departments
                where dname= 'ICS'
            )
MINUS
SELECT salary
FROM lecturers
WHERE dno = (
                SELECT dno
                FROM departments
                WHERE dname= 'COE'
            )
```

Input Tables

Example: Use of INTESECTION

- **List salaries that are taken by both COE and ICS lecturers.**

```
SELECT salary
FROM lecturers
WHERE dno = (
                SELECT dno
                FROM departments
                where dname= 'ICS'
            )
INTERSECTION
SELECT salary
FROM lecturers
WHERE dno = (
                SELECT dno
                FROM departments
                WHERE dname= 'COE'
            )
```

Produces result tables from both queries and creates single result table consisting of those rows that are common to both result tables.

Other SQL Operators

- IN (covered)
- BETWEEN (covered)
- LIKE (covered)
- ANY (SOME)
- ALL
- EXISTS
- NOT EXISTS

ANY (SOME) and ALL

- ANY and ALL may be used with subqueries that produce a single column of numbers.
- If subquery preceded by ALL, condition will only be true if it is satisfied by *all* values produced by subquery.
- If subquery preceded by ANY, condition will be true if it is satisfied by *any* values produced by subquery.
- If subquery is empty, ALL returns true, ANY returns false.
- ISO standard allows SOME to be used in place of ANY.

Example using the SOME Operator

Input Tables

- Find lecturers whose salary higher than the salary of at least 1 COE lecturer.

```
SELECT *
FROM Lecturers
WHERE salary > SOME (
        SELECT salary
        FROM lecturers
        WHERE dno =    (
                SELECT DNO
                FROM department
                WHERE dname = 'COE'
                )
    );
```

Convert previous nested queries by using SOME, ALL

Example Using the ALL Operator

- Find lecturers whose salary higher than the salary of every COE lecturer.

```
SELECT *
FROM Lecturers
WHERE salary > ALL   (
                        SELECT salary
                        FROM lecturers
                        WHERE dno =    (
                                           SELECT DNO
                                           FROM department
                                           WHERE dname = 'COE'
                                       )
                    );
```

EXISTS and NOT EXISTS

- EXISTS and NOT EXISTS are for use only with subqueries specially with **correlated subqueries**. **A correlated subquery** is a subquery where some attributes of the outer select are used in the inner select.
- They produce a simple true/false result.
- EXISTS is true if and only if there exists at least one row in result table returned by subquery.
- It is false if subquery returns an empty result table.
- NOT EXISTS is the opposite of EXISTS.
- Since EXISTS and NOT EXISTS check only for existence or non-existence of rows in subquery result table, subquery can contain any number of columns.

--- Example using the EXISTS Operator

- Find all ICS lecturers.

```
SELECT *
FROM lecturers a
WHERE EXISTS
                (
                    SELECT  1
                FROM   department  b
                WHERE a.dno = b.dno
                    AND    b.dname = 'ICS'
                );
```

Convert previous nested queries by using SOME, ALL

Example using the NOT EXISTS Operator

- Find all non ICS lecturers.

```
SELECT *
FROM lecturers a
WHERE NOT EXISTS
                (
                    SELECT  1
                  FROM   department  b
                  WHERE a.dno = b.dno
                      AND    b.dname = 'ICS'
                );
```

Convert previous nested queries by using EXISTS and NO EXISTS

More SQL Functions

- SUBSTR
- INSTR
- LENGTH
- LEFT, RIGHT
- LPAD, RPAD
- TRIM
- DECODE
- CEIL
- ROWNUM

TO_CHAR

TO_DATE

TO_NUMBER

ADD_MONTHS

FLOOR

SYSDATE

NVL

TRANSLATE

SQL Data Definition Statements (DDL)

- CREATE SCHEMA and DROP SCEHMA +
- CREATE TABLE +
- ALTER TABLE +
- DROP TABLE +

CREATE SCHEMA and DROP SCHEMA

CREATE SCHEMA [name| AUTHORIZATION creator_id];

Example: CREATE USER COMPANY IDENTIFIED BY password;

DROP SCHEMA name [RESTRICT | CASCADE];

Example: DROP USER COMPANY CASCADE;

- With **RESTRICT** (default), schema must be empty or operation fails.
- With **CASCADE**, operation cascades to drop all objects associated with schema in the order defined above. If any of these operations fail, DROP SCHEMA fails.

CREATE TABLE

```
CREATE TABLE table_name
(col_name data_type [NULL | NOT NULL] [,...]);
```

- Creates a table with one or more columns of the specified *data_type*.
- NULL (default) indicates whether column can contain *nulls*.
- With NOT NULL, system rejects any attempt to insert a null in the column.
- Primary keys should always be specified as NOT NULL.
- Foreign keys are often (but not always) candidates for NOT NULL.

CREATE TABLE – Example 1

```
CREATE TABLE Employee
    (
        fname          VARCHAR2(15)        NOT NULL,
        minit          CHAR,
        lname          VARCHAR2(15)        NOT NULL,
        ssn            CHAR(9),
        bdate          DATE,
        address        VARCHAR2(50),
        sex            CHAR,
        salary         NUMBER(10,2)        NOT NULL,
        Superssn       CHAR(9),
        dno            NUMBER(3)           NOT NULL,
      CONSTRAINT employee_ssn_pk PRIMARY KEY(ssn),
      CONSTRAINT employee_superssn_fk
        FOREIGN KEY(Superssn)   REFERENCES employee(ssn),
      CONSTRAINT employee_dno_fk
        FOREIGN KEY(dno) REFERENCES department(dnumber),
    );
```

CREATE TABLE – Example 2

```
CREATE TABLE department
    (
        dname                VARCHAR2(15)        NOT NULL,
        dnumber              NUMBER(3)           NOT NULL,
        mgrssn               CHAR(9),
        mgrStartDateDATE,
   CONSTRAINT department_dnumber_pk
        PRIMARY KEY(dnumber),
   CONSTRAINT department_mgrssn_fk
        FOREIGN KEY(mgrssn) REFERENCES employee(ssn)
    );
```

DROP TABLE

DROP TABLE tbl_name [RESTRICT | CASCADE]

e.g. DROP TABLE employee;

- Removes named table and all rows within it.
- With RESTRICT, if any other objects depend for their existence on continued existence of this table, SQL does not allow request.
- With CASCADE, SQL drops all dependent objects (and objects dependent on these objects).

ALTER TABLE

- The ALTER command is a schema modification command.
- It is used to add or drop a column, change a column definition, add or drop table constraints.

Examples:

```
ALTER TABLE COMPANY.EMPLOYEE
MODIFY(lname VARCHAR2(30));

ALTER TABLE EMP
ADD Constraints pk_emp primary key (EMPNO);

ALTER TABLE EMP
ADD CONSTRAINTS FK_DEPTNO FOREIGN KEY (DEPTNO)
REFERENCES DEPT(DEPTNO);
```

SQL Data Manipulation Statements (DML)

- INSERT Statement +
- UPDATE Statement +
- DELETE Statement +

- Note: Use following control commands for above SQL
- Commit for DO or confirm
- Rollback for UNDO

INSERT Statement

- Definition of INSERT Statement +
- Types of INSERT Statement +
- INSERT and Integrity Constraints +

Definition of INSERT Statement

- INSERT is used to add a single row to a table where we specify the relation name and a list of values for the row.

- There are three types of INSERT Statement:
  - INSERT With Column list +
  - INSERT Without Column list +
  - INSERT with SELECT Statement +

INSERT with Column list

INSERT INTO table_name (column_list) VALUES (data_value_list);

- Example: **INSERT INTO employee(fname, lname, ssn, salary, dno)**
  VALUES ('Majid', 'Al-Ghamdi', '1111111', 4000, 123);

- *data_value_list* must match *column_list* as follows:
  - Number of items in each list must be the same.
  - Must be direct correspondence in position of items in two lists.
  - Data type of each item in *data_value_list* must be compatible with data type of corresponding column.
  - If one of the table columns is omitted from the *column_list* It must also be omitted from the *data_value_list* and make sure it is nullable.

INSERT without Column List

```
INSERT INTO table_name  VALUES (data_value_list);
```

- **Example:**
```
INSERT INTO employee
VALUES ('Adel', NULL, 'Al-Eid', '222222',
NULL, NULL, NULL, NULL, NULL, 1);
```

- *data_value_list* must match the columns of the table as follows:
  - Number of items in the list must be equal to the number of columns of the table.
  - Data type of corresponding items must be compatible.

INSERT … SELECT

- Second form of INSERT allows multiple rows to be copied from one or more tables to another:

```
INSERT INTO table_name [ (column_list) ]
SELECT ...
```

Example:

```
INSERT INTO Table1 (A1, A2, A3)
SELECT B1, B2, B3 FROM Table2;
```

INSERT and Integrity Constraints

- A DBMS that fully implement SQL2 should support and enforce all the integrity constraints that can be specified in the DDL.

- A DBMS enforcing NOT NULL will reject an INSERT command in which an attribute declared to be NOT NULL does not have a value.

- A DBMS not supporting referential integrity will allow insertion even if the referential integrity constraint is violated.

UPDATE

- Definition +
- Examples
  - Update All Rows +
  - Update Specific Rows +
  - Update Multiple Columns +

UPDATE Definition …

- The UPDATE command is used to modify attribute values of one or more selected rows.

```
UPDATE table_name
SET column_name1 = data_value1
    [, column_name2 = data_value2...]
[WHERE search_condition]
```

- *table_name* can be name of a base table or an updatable view.

- SET clause specifies names of one or more columns that are to be updated.

UPDATE Definition …

- WHERE clause is optional:
  - If omitted, named columns are updated for all rows in table.
  - If specified, only those rows that satisfy *search_condition* are updated.

- New *data_value(s)* must be compatible with data type for corresponding column.

Example: UPDATE All Rows

Give all employees a 3% pay increase.

```
UPDATE staff
SET salary = salary*1.03;
```

Example: UPDATE Specific Rows

- Give all Employees in Department one a 5% pay increase.

```
UPDATE employee
SET salary = salary*1.05
WHERE dno = 1;
```

- WHERE clause finds rows that contain data for **dno = 1**. Update is applied only to these particular rows.

Example: UPDATE Multiple Columns

- Change Adel's department to 2 and his Salary to 4,000. Assume Adel's ssn = 111;

```
UPDATE employee
SET dno = 2
    , salary = 4000
WHERE ssn = '111';
```

DELETE

- DELETE Definition +
- DELETE Example +

DELETE Definition

- A DELETE command removes rows from a table and may include a where-clause.

- Rows are explicitly deleted from only one table at a time. However, the deletion may propagate to rows in other tables if referential triggered actions are specified in the referential integrity constraints of the DDL.

DELETE FROM table_name [WHERE search_condition]

- **table_name** can be name of a base table or an updatable view.

- The WHERE clause is optional; if omitted, all rows are deleted from table. But if it is included only those rows that satisfy the **search_condition** are deleted.

Example: DELETE

- Delete all records from employee.

```
DELETE FROM employee;
```

- Delete all employees in department 1.

```
DELETE FROM employee
  WHERE dno = 1;
```

END