

# Database Fundamentals

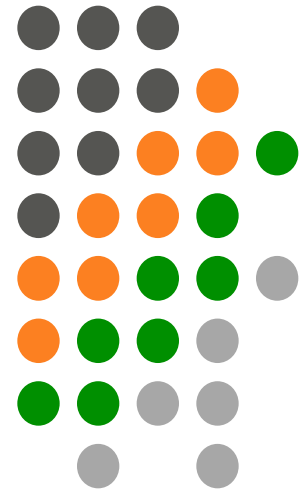
---

## Lecture 2 (More Terminology and SQL)

Lecturer : Dr Irene Murtagh

Room :A15

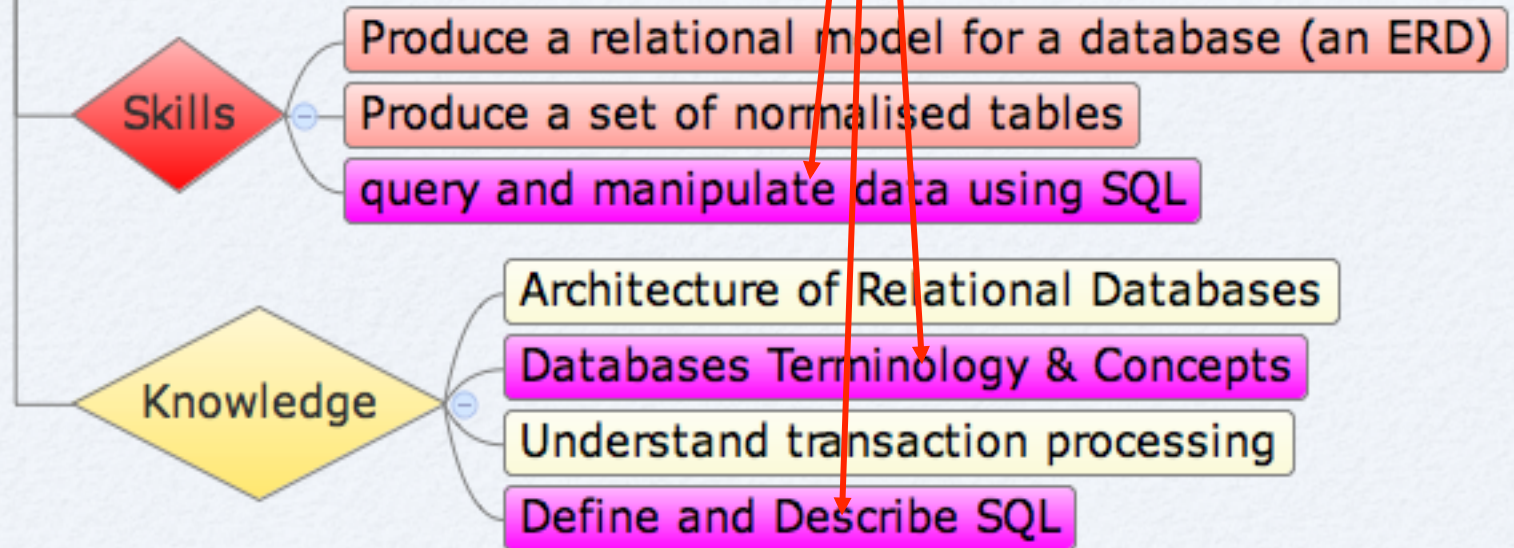
Email: [irene.murtagh@tudublin.ie](mailto:irene.murtagh@tudublin.ie)





# Learning Outcomes

## Databases: Learning Outcomes





# Recap on last week

- What is data?
- What is a database/schema?
- What is a table?
  - What is a row?
  - What is a cell?
- What is a DBMS?
- What is MySQL?





# Objective for this lecture:

1. Cover a little more terminology

2. Start on SQL

- Retrieve data from the database:- the **SELECT** statement.
- The **SELECT** command is used to list the contents of a table



# Terminology

- Four additional terms before we start SQL:
  1. Columns and domains
  2. Primary Key
  3. Foreign Key
  4. Constraints



# 1. More on columns

- Each Column in a database table represents a different attribute.
  - Each column has a distinct name
  - Values in a column must all be of the same type, which is called the **Domain** of the attribute, e.g. **Boolean, date, integer, character** etc.
  - The Domain can also define **VALID VALUES** for each column, e.g. Employee\_Age must be between 16 and 65; Car\_Type must be one of (Ford, Toyota, Fiesta, Audi).
  - Columns contain only **single values** i.e no lists
  - **The Order of columns has no significance.**



## 2. Primary Key:

- Every row in a database table must have some value that UNIQUELY identifies that row, called the **PRIMARY KEY**.
- No two rows in a table can have the same value in the primary key.
- It is usually just one column, but can be made up of more than one column.

StudentID	First Name	Surname	Date of Birth	PhoneNumber
B00012345	John	Murphy	21/10/1980	0851234567
B00009786	Jane	Ryan	04/03/1975	0861298374

What is the primary key in each of these three tables?

ISBN	Book Title	Author
23422345	Introduction to Databases	Colin
43299384	The Boy in the Striped Pyjamas	Boyne, John

Account Number	Customer Number	Balance
23245675	Gray375	100
54366541	Gray375	2500

### 3. *Foreign Key*



- Rows in tables can be linked by having common fields – called **Primary Key / Foreign Key links**
- A **FOREIGN KEY** is an attribute(s), which is the Primary Key in another table, **as per supplier ID in the Parts table on the next slide.**





Tables are linked by having common fields - **Primary Key / Foreign Key** links

Primary Key

Table1: Supplier

Supplier ID	Supplier Name	Supplier Address
S001	Dell	Limerick
S002	Hewlett Packard	London
S003	IBM	Dublin

Table 2: Parts table

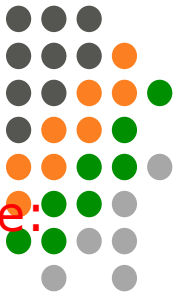
Parts I.D.	Description	Qty on Hand	Supplier I.D.
P001	Keyboard	50	S001
P001	Mouse	100	S001
P003	Printer	25	S003

Foreign Key

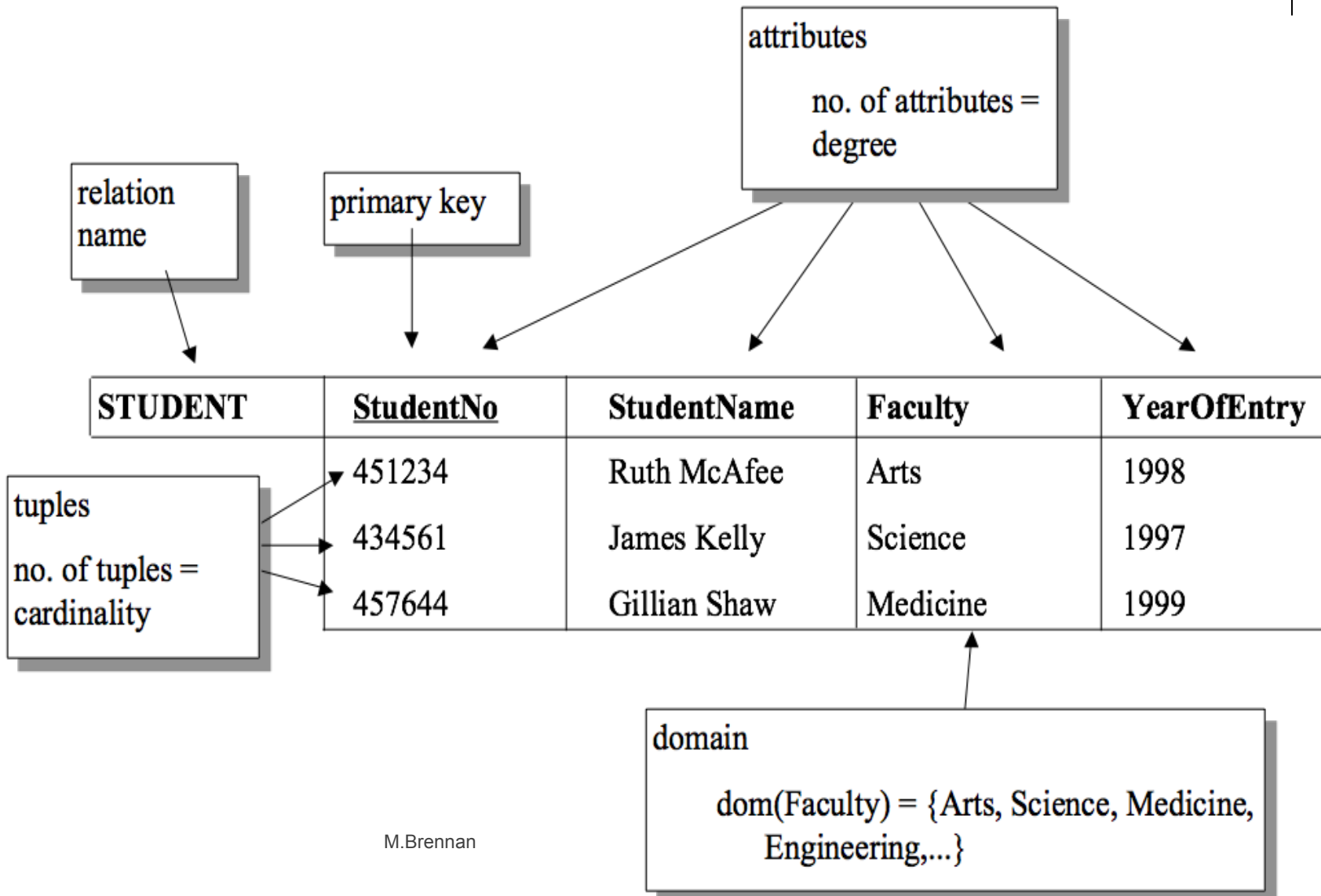
# The Relational DB Model

- Degree of a Table:  
number of columns

- Cardinality of a Table:  
number of rows



STUDENT table



# 4. Table Constraints



- When adding data to a table, the DBMS validates that data in accordance with a number of constraints which apply to data in the table. These constraints fall into four categories:
  - Domain constraints
  - Entity integrity constraint
  - NULL constraint
  - Referential integrity constraint



# Domain Constraints

- The DBMS ensures all data added to a column is valid for the domain of that column. For example:
  - Numeric columns can only contain numeric data
  - Character columns can contain character data
  - If a domain definition puts further restrictions on the data allowed, they will also be checked by the DBMS, for example:
    - Age must be between 16 – 100
    - Sex two valid values: Male, Female
    - Sub. Paid two valid values: True/ False
    - Room four valid values: Single, Double, Family, Twin-bed

# NULL constraint



Only allows a NULL value if a column is allowed to contain NULL values.

Note: **NULL** is different from **blank** or **zero**. Blank is a value, NULL is the absence of a value. Often NULL is used to indicate there is no value for that cell, blank is used if the value is unknown, for example

If an employee table has an attribute **Car\_Registration**. **Null** would be used for employees who do not have a car; a **blank** would be used if the employee does have a car, but the car registration number is not known.

# Entity Integrity Constraint



- The DBMS ensures ALL primary key values are **unique** and **not NULL**.

# Referential Integrity Constraint



- If the foreign key exists in a table, either the foreign key value must be **NULL**, or **match a primary key value** in its home table. For example any value in the supplierID column of the Parts table must be a valid supplier ID in the supplier table.

Table1: Supplier

Supplier ID	Supplier Name	Supplier Address
S001	Dell	Limerick
S002	Hewlett Packard	London
S003	IBM	Dublin

Table 2: Parts file

Parts I.D.	Description	Qty on Hand	Supplier I.D.
P001	Keyboard	50	S001
P001	Mouse	100	S001
P003	Printer	25	S003

# Question - Time





# Do you remember what the following terms mean?



Referential  
integrity

Tuple

Entity  
integrity

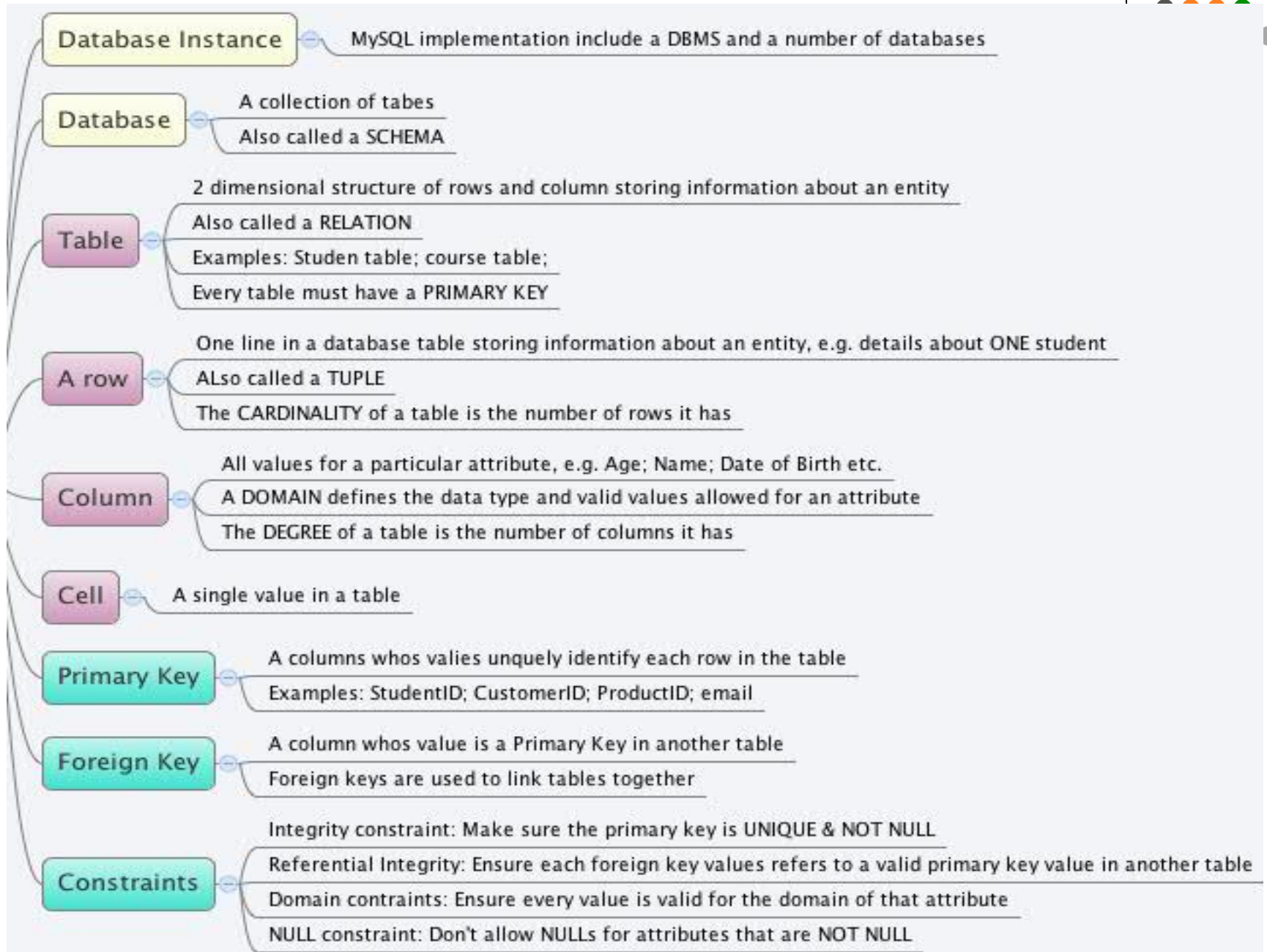
Null  
Constraint

A relation

Schema

Domain

## Recap - terminology





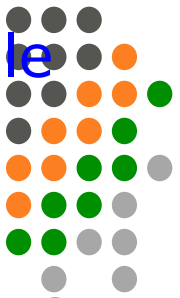
# SQL – STRUCTURED QUERY LANGUAGE

# Introduction to SQL



- Universal Language for
  - Creating Tables to hold the data (Data Definition Language – DDL: 6 commands)
  - Data Manipulation & Retrieval (Data Manipulation Language – DML: 8 commands)
  - Data Control – gives users permissions for the database (Data Control Language – DCL: 3 commands)
- Note: While SQL is a standard language, Database vendors support slight variations of SQL. Variations occur in the data types supported, and the functions support (to be covered in a later lecture)

DDL (Data Definition Language) used to define the table structure and attributes of the database table



SQL commands:-

- CREATE TABLE specifies attributes and constraints for a table.
- DROP TABLE
- ALTER TABLE
- TRUNCATE etc.

DML (Data Manipulation Language) used to **retrieve**, insert, modify or delete information within the database.

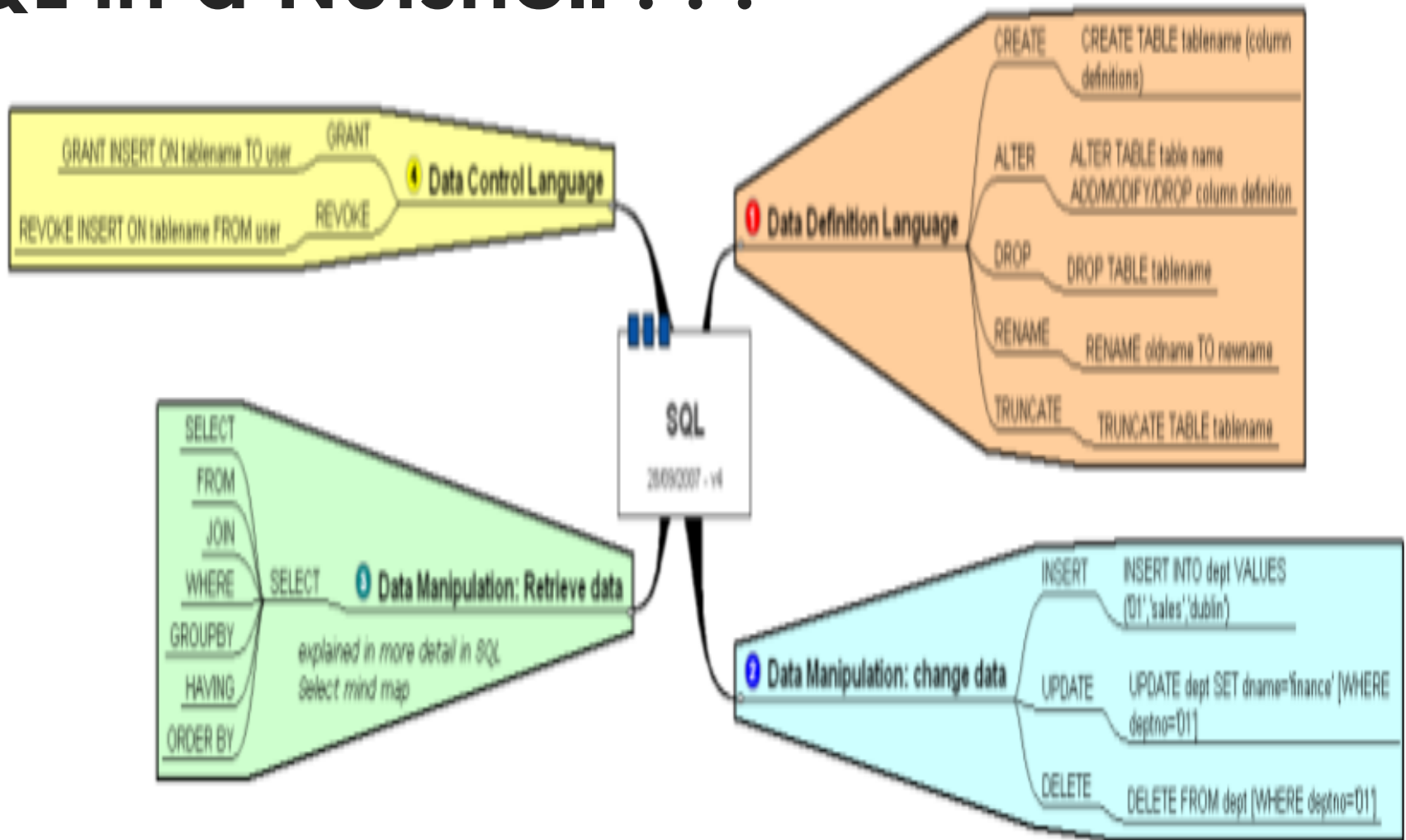
SQL commands:- SELECT, UPDATE, INSERT, DELETE

DCL (Data Control Language) - used to manage DB security, i.e. assign access rights to users

SQL commands:- GRANT, DENY, REVOKE



# SQL in a Nutshell . . .



# Sample Tables

- The slides in this section of the course are based on the following tables:



Employee table - 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		30
7788	SCOTT	ANALYST	7566	09-DEC-82	3000		20
7839	KING	PRESIDENT		17-NOV-81	5000		10
7844	TURNER	SALESMAN	7698	08-SEP-81	1500	0	30
7876	ADAMS	CLERK	7788	12-JAN-83	1100		20
7900	JAMES	CLERK	7698	03-DEC-81	950		30
7902	FORD	ANALYST	7566	03-DEC-81	3000		20
7934	MILLER	CLERK	7782	23-JAN-82	1300		10


Department table - DEPT

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

# Data Retrieval – SELECT Statement

- Three basic commands
  - Selecting rows
  - Selecting columns
- Join – get results from 2 or more tables

Selection




Projection


Join


Table 1


Table 2



# Basic SELECT examples



Display the name of each employee

SELECT ename

Column Name

FROM emp;

Table Name

Employee table - 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		30
7788	SCOTT	ANALYST	7566	09-DEC-82	3000		20
7839	KING	PRESIDENT		17-NOV-81	5000		10
7844	TURNER	SALESMAN	7698	08-SEP-81	1500	0	30
7876	ADAMS	CLERK	7788	12-JAN-83	1100		20
7900	JAMES	CLERK	7698	03-DEC-81	950		30
7902	FORD	ANALYST	7566	03-DEC-81	3000		20
7934	MILLER	CLERK	7782	23-JAN-82	1300		10

output

ENAME  
-----  
SMITH  
ALLEN  
WARD  
JONES  
MARTIN  
BLAKE  
CLARK  
SCOTT  
KING  
TURNER  
ADAMS  
JAMES  
FORD  
MILLER

The table DOES NOT change, you are just selecting what data you want to view from the table.

# Basic SELECT examples

Display the employee number, name and salary of each employee

```
SELECT empno, ename, sal
```

Column Names

```
FROM emp;
```

Table Name

Employee table - 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		30
7788	SCOTT	ANALYST	7566	09-DEC-82	3000		20
7839	KING	PRESIDENT		17-NOV-81	5000		10
7844	TURNER	SALESMAN	7698	08-SEP-81	1500	0	30
7876	ADAMS	CLERK	7788	12-JAN-83	1100		20
7900	JAMES	CLERK	7698	03-DEC-81	950		30
7902	FORD	ANALYST	7566	03-DEC-81	3000		20
7934	MILLER	CLERK	7782	23-JAN-82	1300		10

output

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



# Basic SELECT examples

List all employee's names, positions and date they were hired.

```
SELECT ename, job, hiredate  
FROM emp;
```

Column Names

Table Name

Employee table - 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		30
7788	SCOTT	ANALYST	7566	09-DEC-82	3000		20
7839	KING	PRESIDENT		17-NOV-81	5000		10
7844	TURNER	SALESMAN	7698	08-SEP-81	1500	0	30
7876	ADAMS	CLERK	7788	12-JAN-83	1100		20
7900	JAMES	CLERK	7698	03-DEC-81	950		30
7902	FORD	ANALYST	7566	03-DEC-81	3000		20
7934	MILLER	CLERK	7782	23-JAN-82	1300		10

output

ENAME	JOB	HIREDATE
SMITH	CLERK	17-DEC-80
ALLEN	SALESMAN	20-FEB-81
WARD	SALESMAN	22-FEB-81
JONES	MANAGER	02-APR-81
MARTIN	SALESMAN	28-SEP-81
BLAKE	MANAGER	01-MAY-81
CLARK	MANAGER	09-JUN-81
SCOTT	ANALYST	09-DEC-82
KING	PRESIDENT	17-NOV-81
TURNER	SALESMAN	08-SEP-81
ADAMS	CLERK	12-JAN-83
JAMES	CLERK	03-DEC-81
FORD	ANALYST	03-DEC-81
MILLER	CLERK	23-JAN-82



List all data in the employee table.

```
SELECT *  
FROM emp;
```

An \* is short hand for  
listing ALL columns in the  
table

List the department numbers in the employee table

```
SELECT deptno  
FROM emp;
```

Display the departments that employees work in. Do not  
show duplicate department numbers

```
SELECT DISTINCT deptno  
FROM emp;
```

output

Deptno
-----
10
20
30

# Selecting Columns



- ◉ The syntax of the most basic form of SELECT is:

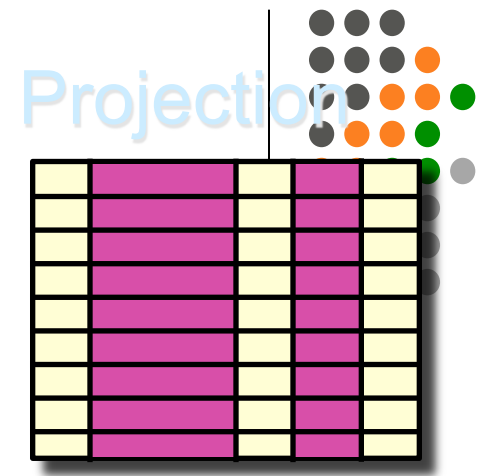
```
SELECT [DISTINCT] {*, column_name [alias], ...}  
FROM table;
```

- ◉ **SELECT** specifies the columnlist / attributes, separated by commas
- ◉ **DISTINCT** – eliminates duplicate rows in resultset
- ◉ **\*** (asterisk) is a wildcard character to list all attributes
- ◉ **FROM** specifies the table



## Note:-

- SQL statements are not case sensitive
- Programming convention is to show all reserved words in uppercase
- Statements can be on one or more lines
- Can use tabs to enhance readability
- Clauses are usually put on a new line
- Use of the semicolon at the end of a SQL statement is optional



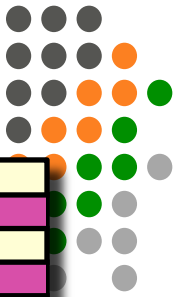
# Exercises

- List all **locations** on the **department** table
- List all **salaries** on the **employee** table
- List all **salaries**, but do not show **duplicate** salaries
- List each **employee** by name, and also show their **salary**
- List each **employee** by name, showing **salary** and **commission** as well.
- Show **all columns** on the **department** table

## 2. Selecting particular rows

- To select particular rows, you add a 'WHERE' clause to the SELECT statement specifying which rows to select






# Where condition

- The SELECT statement retrieves all the rows that **match** the conditions you specified in **the WHERE clause**
  - You can have more than one condition, separated by logical operators (AND, OR, NOT)
- If no rows match the criteria specified in the WHERE clause, you get a message that tells you that no rows were returned.

## Examples of WHERE clause

- Display the name, job title and department numbers of all employees **who are clerks**.

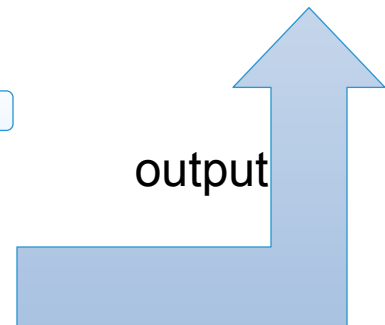
```
SELECT ename, job, deptno  
FROM emp  
WHERE job='CLERK';
```

<u>ENAME</u>	<u>JOB</u>	<u>DEPTNO</u>
JAMES	CLERK	10
SMITH	CLERK	20
ADAMS	CLERK	20
MILLER	CLERK	30



Employee table - 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		30
7788	SCOTT	ANALYST	7566	09-DEC-82	3000		20
7839	KING	PRESIDENT		17-NOV-81	5000		10
7844	TURNER	SALESMAN	7698	08-SEP-81	1500	0	30
7876	ADAMS	CLERK	7788	12-JAN-83	1100		20
7900	JAMES	CLERK	7698	03-DEC-81	950		30
7902	FORD	ANALYST	7566	03-DEC-81	3000		20
7934	MILLER	CLERK	7782	23-JAN-82	1300		10





**Note:** Character strings and dates  
are enclosed in single quotes

# Where Conditions



- ◉ Comparison Operators:
  - > =, >, >=, <, <=, <> (not equal to), != (not equal to), IS NULL, IS NOT NULL
- ◉ Ranges
  - > BETWEEN, NOT BETWEEN
- ◉ Lists
  - > IN, NOT IN
- ◉ Character Matches
  - > LIKE, NOT LIKE
- ◉ Combinations of the above
  - > AND, OR, NOT
  - > AND evaluates to true if all expressions are true
  - > OR evaluates to true if any of the expressions are true
  - > NOT evaluates to false if the expression is true and true if the expression is false.

# Examples – comparison operators



Display the names and salaries for employees earning at least 1500

SELECT ename, sal

FROM emp

WHERE sal >= 1500;

Employee table - 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		30
7788	SCOTT	ANALYST	7566	09-DEC-82	3000		20
7839	KING	PRESIDENT		17-NOV-81	5000		10
7844	TURNER	SALESMAN	7698	08-SEP-81	1500	0	30
7876	ADAMS	CLERK	7788	12-JAN-83	1100		20
7900	JAMES	CLERK	7698	03-DEC-81	950		30
7902	FORD	ANALYST	7566	03-DEC-81	3000		20
7934	MILLER	CLERK	7782	23-JAN-82	1300		10



# Examples – comparison operators

Display the names and salaries of all employees whose monthly salary is less than or equal to the commission they earn

```
SELECT ename, sal  
FROM emp  
WHERE sal <= comm;
```

List all employees who are not clerks

```
SELECT empno, ename  
FROM emp  
WHERE job <> 'clerk'
```



# Example – checking for NULL

Which employees are NOT on Commission?

```
SELECT ename  
FROM emp  
WHERE comm IS NULL;
```

Which employees are on Commission?

```
SELECT ename  
FROM emp  
WHERE comm IS NOT NULL;
```



# Examples – BETWEEN (for numeric data & dates)

List the names of employees who earn between 1000 and 1500 a month.

```
SELECT ename  
FROM emp  
WHERE sal BETWEEN 1000 AND 1500;
```

Show all details for employees hired during 1982 (i.e. between Jan 1<sup>st</sup> 1982 and Dec 31<sup>st</sup> 1982)

```
SELECT *  
FROM emp  
WHERE hiredate BETWEEN '1982-01-01' AND '1982-12-31'
```

BETWEEN checks for an inclusive range.

Date format is  
**YYYY-MM-DD**  
Year, month, day



## Examples – IN, NOT IN

check if a value is one of a list of values



Give employee details for employees whose manager number is 7902, 7566, or 7788

```
SELECT empno, ename  
FROM emp  
WHERE mgr IN (7902, 7566, 7788);
```

Numeric Data

Return all employees who are not managers, clerks or salesmen.

```
SELECT empno, ename, job  
FROM emp  
WHERE job NOT IN ('manager', 'clerk', 'salesman');
```

Character Data

# Exercises



- Write SQL statements for the following:
  - display the name and salary of all employees earning more than \$2850
  - display the name and department for employee number 7566
  - display the name, job and start date for employees hired between Feb 20th 1981 and May 1st, 1981.

**Select . . .** (which columns do you want to show in the output?)

**From . . .** (what is the name of the table?)

**Where . . .** (which rows should be displayed in the output?)

**Exercise** – using a different table.  
**Table name: student**



StudentID	Surname	Course	Age	County
B00012345	Murphy	BN002	19	Dublin 15
B00034593	Doyle	BN104	34	Meath
B00043894	Hope	BN002	27	Dublin 7
B00345640	Keily	BN103	19	Dublin 15

- Which students are enrolled in the Certificate in Computing (BN002)?
- We need to send a mail shot to students not living in Dublin 15. List all such students.
- Are any students under 18?

## Wildcard searches using LIKE (for strings - varchar)



- ◉ Use LIKE operator to find **patterns** in **strings (varchar)**. Typically look for a specific pattern in the values of the rows in a given table
- ◉ Condition can contain
  1. characters or number
  2. % denotes a string of any length (0 or more)
  3. \_ (underscore) denotes any single character
  4. search for a character in a range [a–e] (**any character between a and e**)
  5. search for a set of characters [abc] (**character must be either a, b or c**)

**Get all employees whose name begins with the letter S**

SELECT ename

FROM emp

WHERE ename LIKE 'S%';



**Get all employees whose name ends with the letter S**

SELECT ename

FROM emp

WHERE ename LIKE '%S';

**Get all employees whose name contains the character sequence **ae****

SELECT ename

FROM emp

WHERE ename LIKE '%ae%';



Find all employees whose name begins with any character and the last four characters are **anet** ( **5 letters in name** )

```
SELECT ename
```

```
FROM emp
```

```
WHERE ename LIKE '_anet';
```

Find all employees whose name begins either with J or S

```
SELECT ename
```

```
FROM emp
```

```
WHERE ename LIKE '[js]%' ;
```



# Limitations of MySQL

- **Note:** currently MySQL only supports using % and \_ with the like operator, but **not** character ranges such as [a-e]. To do more complex pattern matching with MySQL, you need to use the **REGEXP** operator instead of **LIKE**. This will interpret patterns based on Java's Regular Expression (REGEX) syntax. This is outside the scope of the module, but as an example, a query looking for all employee names beginning with J or S would be written as:

Select ename

From emp

Where ename REGEXP '^[JS]'



# Exercises

- List all employees working in a department whose number starts with the digits '76'
- Which employees have the letter **A** in their name?
- Which employees have names that start with a letter from the first half of the alphabet (**A to M**)?
- Which names start with the letters **B** or **C**?





# Exercise – Product table of furniture available in IKEA

ProductID	Description	Range	Aisle	Section
10203097	TV Bench	BENNO	18	A
30133942	TV Bench with Panel	BENNO	18	B
70104438	TV Bench	<b>GREVBÄCK</b>	18	F
60105339	Corner TV Bench	<b>LACK</b>	18	D
10065958	Coffee Table	LACK	15	H
07305310	DVD Tower	BENNO	2	J

- Based on the table above, write queries for the following:
  - List all products in the BENNO range
  - List all products in Aisle's 15 or higher
  - List all products in sections A, B and C
  - List all the types of TV Benches available



# Combining conditions

Where clauses can include more than one condition, conditions can be joined using logical operators: **AND, OR**

Operators are evaluated in the following order - **Rules of Precedence**

1. Comparison Operators
2. NOT
3. AND
4. OR

Use parentheses / brackets to override rules



# Example – combining conditions

Display the employee number, name, position and salary of all employees earning at least 1100 per month employed as clerks.

```
SELECT empno, ename, job, sal
FROM emp
WHERE sal >= 1100
AND job = 'CLERK';
```

Rows meeting both search conditions are returned.



# Example – combining conditions

List the name, job and salary for staff who are employed either as salesmen or presidents and who earn more than 1500 a month.

```
SELECT ename, job, sal
FROM emp
WHERE job='SALESMAN'
OR job='PRESIDENT'
AND sal>1500;
```



## Example: Combining conditions

- Return all employees whose name begins with 'b' and are not employed as clerks, presidents or managers

```
SELECT empno, ename, job  
FROM emp  
WHERE ename LIKE 'b%'  
AND job NOT IN ('clerk', 'manager', 'president');
```



# Exercise

- Display the name, job, and salary for all employees whose job is Clerk or Analyst and their salary is not equal to \$1000, \$3000, or \$5000.
- Display the name of all employees who have 'LL' in their name and are in department 30 or their manager is 7782.
  - Is the English ambiguous?



# Derived Columns

- You can derive a new column from existing numeric or date fields using arithmetic expressions as shown below. This does not make any change to the database table itself.

```
SELECT ename, sal, (sal *12) +100 AS 'Annual Salary'  
FROM emp;
```

ENAME	SAL	Annual Salary
KING	5000	60100
BLAKE	2850	34300
CLARK	2450	29500
JONES	2975	35800
MARTIN	1250	15100
ALLEN	1600	19300
...		

Operator  
Precedence:

$*, /, +, -$

# Renaming Column Headings



- In the last slide, the new column was display under a column heading of 'Total Salary'.
- The display name of a column can be changed in three ways:

```
SELECT ename AS NAME, sal SALARY, sal*12 AS 'Total  
Salary' FROM emp;
```

1. With **AS**

2. Without **AS**

3. If the new name includes  
a blank, then it must be in  
**quotes**

NAME	SALARY	Total Salary
-----	-----	-----
KING	5000	60000
BLAKE	2850	34200
CLARK	2450	29400
JONES	2975	35700





# Adding text to the output

- Characters, numbers or dates can be outputted as part of each row:

```
SELECT ename,' is a ', job AS 'Employee Details'  
FROM   emp;
```

ename	is a	Employee Details
KING	is a	PRESIDENT
BLAKE	is a	MANAGER
CLARK	is a	MANAGER
JONES	is a	MANAGER
MARTIN	is a	SALESMAN
ALLEN	is a	SALESMAN
TURNER	is a	SALESMAN
JAMES	is a	CLERK



# Sorting rows for output

- **ORDER BY** clause SORTS output, and should be the **last** clause in a SELECT statement

```
SELECT [DISTINCT] {*, column_name [alias], ...}  
FROM table  
[WHERE condition(s)];  
[ORDER BY {column, expression} [ASC | DESC]];
```

The default sort method is Ascending.



# Sorting rows for output

Display each employee's name, position, department and the date they were hired. Sort the output in ascending order according to hiredate.

```
SELECT ename, job, deptno, hiredate
FROM   emp
ORDER BY hiredate;
```

ENAME	JOB	DEPTNO	HIREDATE
SMITH	CLERK	20	17-DEC-80
ALLEN	SALESMAN	30	20-FEB-81
...			

14 rows selected.

# Sorting rows for output



Display each employee's name, position, department and the date they were hired. Sort the by date, with the most recently employed listed first.

```
SELECT ename, job, deptno, hiredate
FROM emp
ORDER BY hiredate DESC;
```

ENAME	JOB	DEPTNO	HIREDATE
ADAMS	CLERK	20	12-JAN-83
SCOTT	ANALYST	20	09-DEC-82
MILLER	CLERK	10	23-JAN-82
JAMES	CLERK	30	03-DEC-81
FORD	ANALYST	20	03-DEC-81
KING	PRESIDENT	10	17-NOV-81
MARTIN	SALESMAN	30	28-SEP-81
...			

14 rows selected.

# Using multiple expressions in the ORDER BY clause



Display each employee's name, position, department and the date they were hired. Sort by job in alphabetical order with the most recently employed in each job category listed first.

```
SELECT ename, job,  
       deptno, hiredate  
FROM emp  
ORDER BY job ASC,  
       hiredate DESC;
```

ename	job	deptno	hiredate
SCOTT	ANALYST	20	1982-12-0
FORD	ANALYST	20	1981-12-0
ADAM	CLERK	20	1983-01-1
MILLE	CLERK	10	1982-01-2
JAMES	CLERK	30	1981-12-0
SMITH	CLERK	20	1980-12-1
CLARK	MANAGER	10	1981-06-0
BLAKE	MANAGER	30	1981-05-0
JONES	MANAGER	20	1981-04-0



# How will the results of the following query be ordered?

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

ename	job	deptno	sal
JAMES	CLERK	30	950
WARD	SALESMAN	30	1250
MARTIN	SALESMAN	30	1250
TURNER	SALESMAN	30	1500
ALLEN	SALESMAN	30	1600
BLAKE	MANAGER	30	2850
SMITH	CLERK	20	800
ADAMS	CLERK	20	1100
JONES	MANAGER	20	2975
FORD	ANALYST	20	3000
SCOTT	ANALYST	20	3000
MILLER	CLERK	10	1300
CLARK	MANAGER	10	2450
KING	PRESIDENT	10	5000



# Limit (or Top N) clause

- ◉ LIMIT is used to limit the number of rows returned by a query as follows

```
SELECT ename, sal  
FROM emp  
ORDER BY sal DESC  
LIMIT 10;
```







# Exercises

1. Give a list of the name and salary of each employee earning more than \$1500, and working in department 10 or 30. Label the columns 'Employee' and 'Monthly Salary'
2. Show a list of unique salary values from the employee table in descending sequence
3. Show the top 3 salary values.
4. List each employee's name, department, and their total earnings for the year where total earnings is  $(\text{sal} * 12 + \text{comm})$ . Name the columns 'Employee', 'Department' and 'Yearly Earnings'

# Summary



## Select overview

Projection – select columns

Selection: select rows

Join: join columns

## Select column:

`SELECT [DISTINCT] {*, column_name [alias], ...} FROM table name`

`SELECT empno, name FROM emp`

Derived columns: `SELECT ename, sal*12 AS 'total salary' FROM emp`

## Select rows:

add a WHERE clause, examples below

`WHERE ename = 'SCOTT'`

`Where sal > 2000`

`where ename like '%T'`

`where sal between 1000 and 2000`

`where detno in [20,30] AND sal > 1500`

## Sorting output

ORDER BY clause – must be last clause

`SELECT ename FROM emp ORDER BY sal`

## TOP N

`select TOP 10 ename FROM emp ORDER BY sal`