8.3 Analyzes the main components of a database system

Time: 14 periods

Learning Outcomes

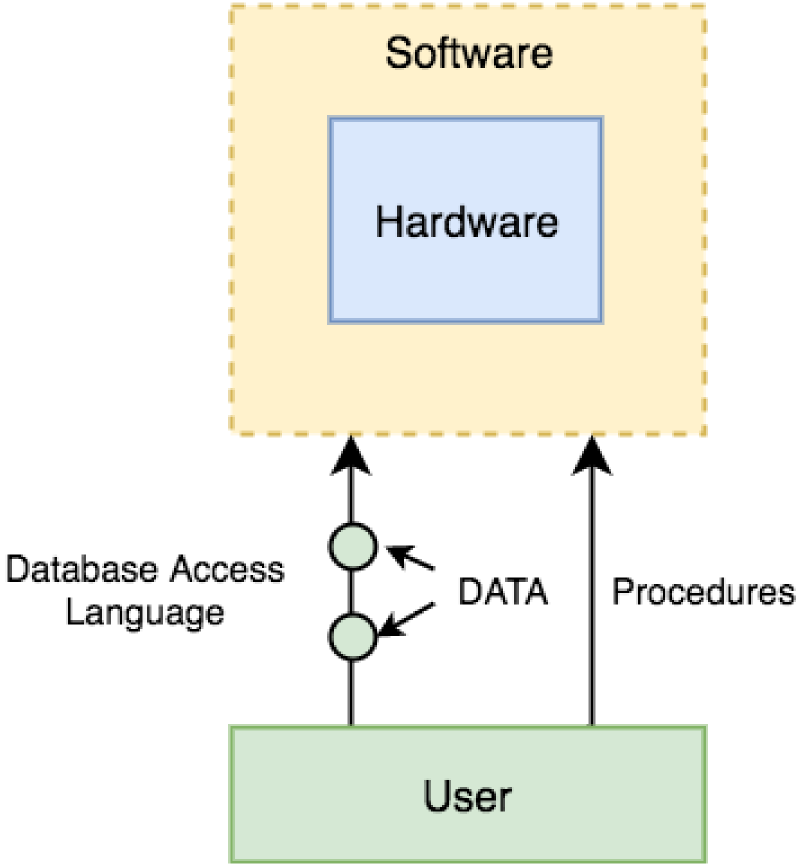
- Lists and briefly describes the component of a database system
- Describes the database management system
- Defines SQL
- Distinguishes between DDL vs. DML
- Uses appropriate SQL commands for creating and using database
- Uses appropriate commands to create tables with suitable fields and data types
- Sets primary key and foreign key while creating table
- Uses primary key and foreign key after completion of a table
- Creates relationships among tables
- Uses appropriate SQL commands to Insert and delete columns, delete foreign key / primary key and to drop table
- Uses appropriate SQL commands to drop database
- Uses appropriate commands to Insert, modify retrieve, update and delete data.
- Uses appropriate DML commands to query data according to the requirements

Components of DBMS

The database management system can be divided into five major components, they are:

- Hardware
- Software
- Data
- Procedures
- Database Access Language

Let's have a simple diagram to see how they all fit together to form a database management system.



Hardware

When we say Hardware, we mean computer, hard disks, I/O channels for data, and any other physical component involved before any data is successfully stored into the memory.

When we run Oracle or MySQL on our personal computer, then our computer's Hard Disk, our Keyboard using which we type in all the commands, our computer's RAM, ROM all become a part of the DBMS hardware.

Software

This is the main component, as this is the program which controls everything. The DBMS software is more like a wrapper around the physical database, which provides us with an easy-to-use interface to store, access and update data.

The DBMS software is capable of understanding the Database Access Language and interpret it into actual database commands to execute them on the DB.

Data

Data is that resource, for which DBMS was designed. The motive behind the creation of DBMS was to store and utilize data.

In a typical Database, the user saved Data is present and Meta data is stored.

Metadata is data about the data. This is information stored by the DBMS to better understand the data stored in it.

For example: When you store your Name in a database, the DBMS will store when the name was stored in the database, what is the size of the name, is it stored as related data to some other data, or is it independent, all this information is metadata.

Procedures

Procedures refer to general instructions to use a database management system. This includes procedures to setup and install a DBMS, to login and logout of DBMS software, to manage databases, to take backups, generating reports etc.

Database Access Language

Database Access Language is a simple language designed to write commands to access, insert, update and delete data stored in any database.

A user can write commands in the Database Access Language and submit it to the DBMS for execution, which is then translated and executed by the DBMS.

User can create new databases, tables, insert data, fetch stored data, update data and delete the data using the access language.

Users

Database Administrators:

Database Administrator or DBA is the one who manages the complete database management system. DBA takes care of the security of the DBMS, it's availability, managing the license keys, managing user accounts and access etc.

Application Programmer or Software Developer:

This user group is involved in developing and designing the parts of DBMS.

End User:

These days all the modern applications, web or mobile, store user data. End users are the one who store, retrieve, update and delete data.

What is a Database Management System?

A DBMS is a technology tool that directly supports data management. It is a package designed to define, manipulate, and manage data in a database.

Popular DBMS Software

Here, is the list of some popular DBMS system:

MySQL

Microsoft Access

Oracle

PostgreSQL

dBASE

FoxPro

SQLite

IBM DB2

LibreOffice Base

MariaDB

Microsoft SQL Server etc.

Application of DBMS

Sector	**Use of DBMS**
Banking	For customer information, account activities, payments, deposits, loans, etc.
Airlines	For reservations and schedule information.
Universities	For student information, course registrations, colleges and grades.
Telecommunication	It helps to keep call records, monthly bills, maintaining balances, etc.
Finance	For storing information about stock, sales, and purchases of financial instruments like stocks and bonds.
Sales	Use for storing customer, product & sales information.
Manufacturing	It is used for the management of supply chain and for tracking production of items. Inventories status in warehouses.
HR Management	For information about employees, salaries, payroll, deduction, generation of paychecks, etc.

SQL

- SQL stands for Structured Query Language
- SQL lets you access and manipulate databases
- SQL became a standard of the American National Standards Institute (ANSI) in 1986, and of the International Organization for Standardization (ISO) in 1987

What Can SQL do?

- SQL can
  - execute queries against a database
  - retrieve data from a database
  - insert records in a database
  - update records in a database
  - delete records from a database
  - create new databases
  - create new tables in a database
  - create stored procedures in a database
  - create views in a database
  - set permissions on tables, procedures, and views

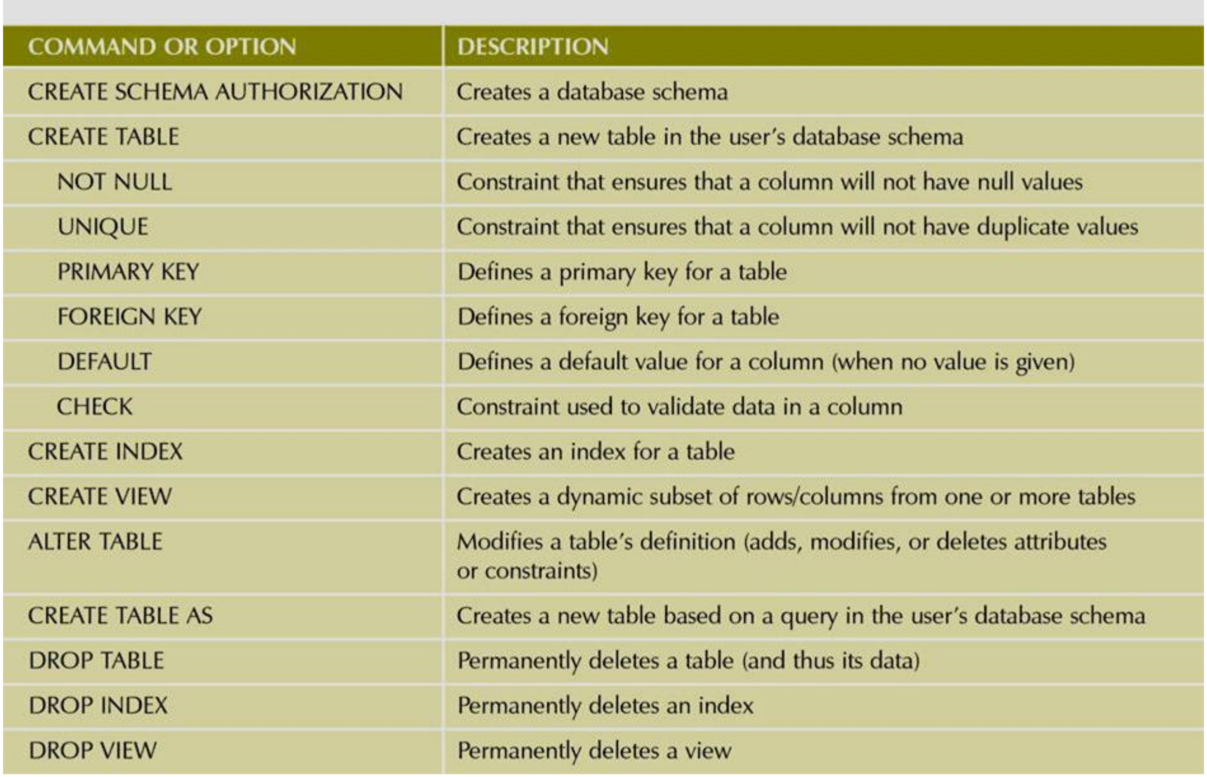
SQL Commands fall into two categories

1. Data Definition Language (DDL)
2. Data Manipulation Language (DML)

Data Definition Language (DDL)

- It is a language used for defining and modifying the data and its structure.
- It is used to build and modify the structure of your tables and other objects in the database.

- **DDL commands are as follows,**

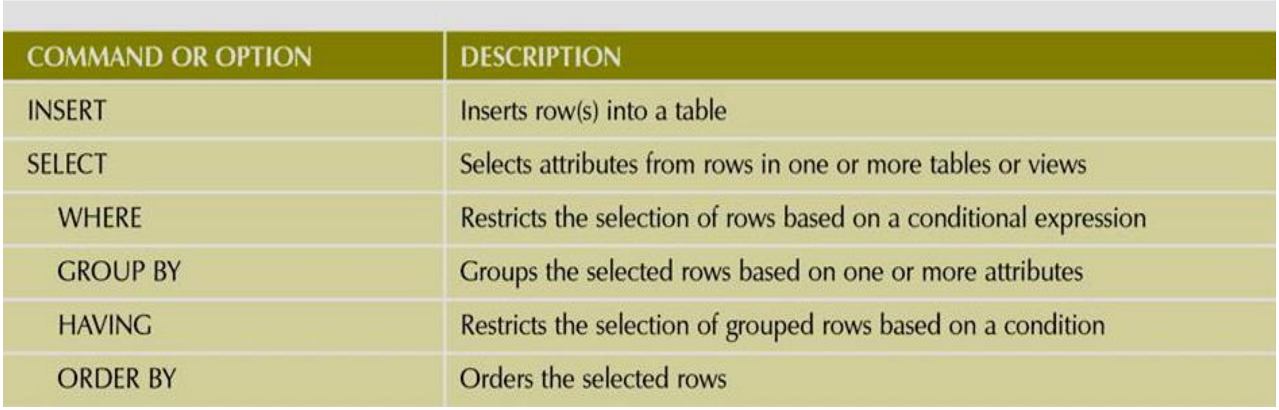


COMMAND OR OPTION	DESCRIPTION
CREATE SCHEMA AUTHORIZATION	Creates a database schema
CREATE TABLE	Creates a new table in the user's database schema
NOT NULL	Constraint that ensures that a column will not have null values
UNIQUE	Constraint that ensures that a column will not have duplicate values
PRIMARY KEY	Defines a primary key for a table
FOREIGN KEY	Defines a foreign key for a table
DEFAULT	Defines a default value for a column (when no value is given)
CHECK	Constraint used to validate data in a column
CREATE INDEX	Creates an index for a table
CREATE VIEW	Creates a dynamic subset of rows/columns from one or more tables
ALTER TABLE	Modifies a table's definition (adds, modifies, or deletes attributes or constraints)
CREATE TABLE AS	Creates a new table based on a query in the user's database schema
DROP TABLE	Permanently deletes a table (and thus its data)
DROP INDEX	Permanently deletes an index
DROP VIEW	Permanently deletes a view

Data Manipulation Language (DML)

- It is a language used for selecting, inserting, deleting and updating data in a database.
- It is used to retrieve and manipulate data in a relational database.

- **DML commands are as follows,**



COMMAND OR OPTION	DESCRIPTION
INSERT	Inserts row(s) into a table
SELECT	Selects attributes from rows in one or more tables or views
WHERE	Restricts the selection of rows based on a conditional expression
GROUP BY	Groups the selected rows based on one or more attributes
HAVING	Restricts the selection of grouped rows based on a condition
ORDER BY	Orders the selected rows

- DML performs read-only queries of data.

DML Query Examples

Selecting Rows with Conditions

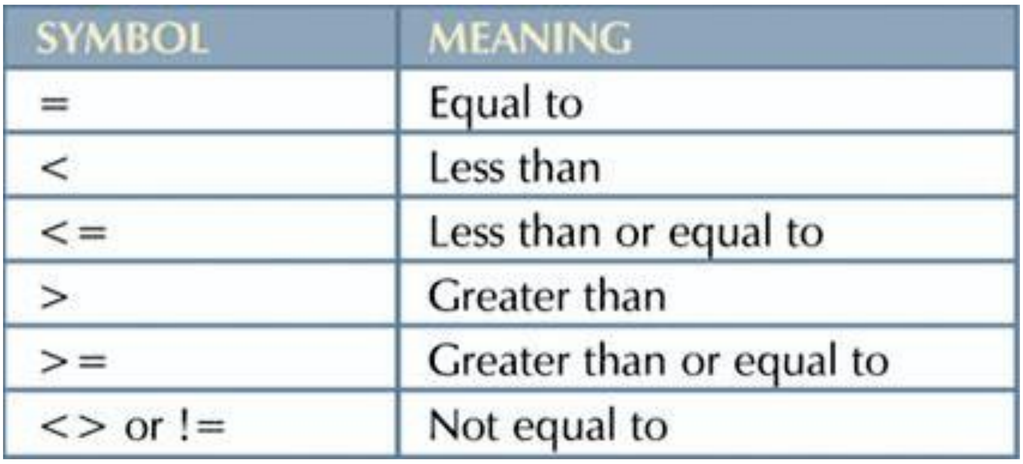
- **Select partial table contents** by placing restrictions on rows to be included in output
  - Add conditional restrictions to SELECT statement, using WHERE clause
  - Syntax:

SELECT column-list

FROM table-list

[WHERE condition-list];

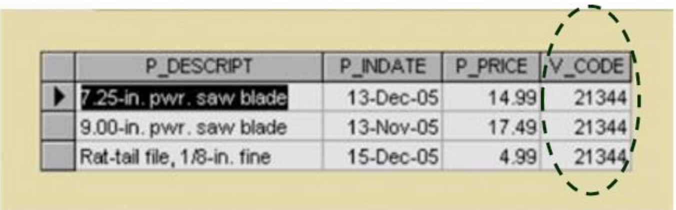
Comparison Operators for Conditional Restrictions



SYMBOL	MEANING
=	Equal to
<	Less than
<=	Less than or equal to
>	Greater than
>=	Greater than or equal to
<> or !=	Not equal to

Example 1

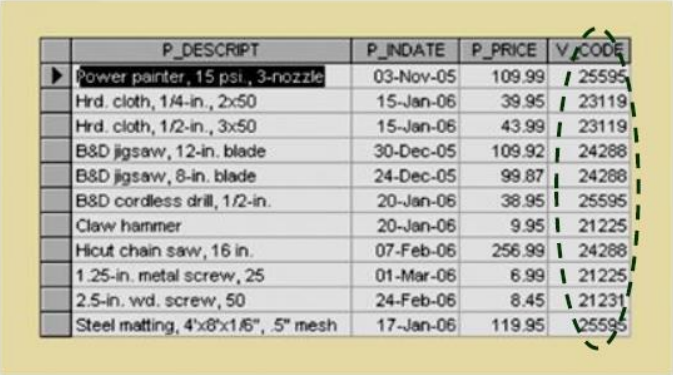
```
SELECT P_DESCRIP, P_INDATE, P_PRICE, V_CODE
FROM PRODUCT
WHERE V_CODE = 21344;
```



P_DESCRIPT	P_INDATE	P_PRICE	V_CODE
7.25-in. pwr. saw blade	13-Dec-05	14.99	21344
9.00-in. pwr. saw blade	13-Nov-05	17.49	21344
Rat-tail file, 1/8-in. fine	15-Dec-05	4.99	21344

Example 2

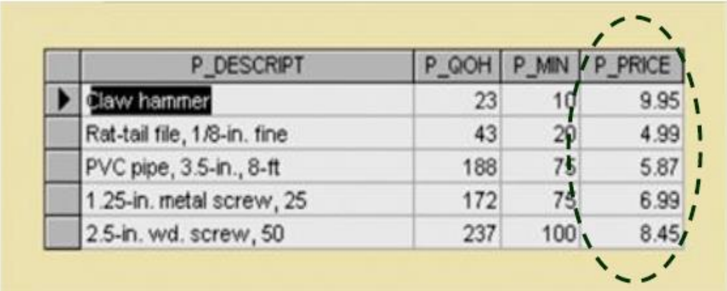
```
SELECT P_DESCRIP, P_INDATE, P_PRICE,
V_CODE
FROM PRODUCT
WHERE V_CODE <> 21344;
```



P_DESCRIPT	P_INDATE	P_PRICE	V_CODE
Power painter, 15 psi., 3-nozzle	03-Nov-05	109.99	25595
Hrd. cloth, 1/4-in., 2x50	15-Jan-06	39.95	23119
Hrd. cloth, 1/2-in., 3x50	15-Jan-06	43.99	23119
B&D jigsaw, 12-in. blade	30-Dec-05	109.92	24288
B&D jigsaw, 8-in. blade	24-Dec-05	99.87	24288
B&D cordless drill, 1/2-in.	20-Jan-06	38.95	25595
Claw hammer	20-Jan-06	9.95	21225
Hicut chain saw, 16 in.	07-Feb-06	256.99	24288
1.25-in. metal screw, 25	01-Mar-06	6.99	21225
2.5-in. wd. screw, 50	24-Feb-06	8.45	21231
Steel matting, 4'x8'x1/6", .5" mesh	17-Jan-06	119.95	25595

Example 3

```
SELECT P_DESCRIP, P_QOH, P_MIN, P_PRICE
FROM PRODUCT
WHERE P_PRICE <= 10;
```

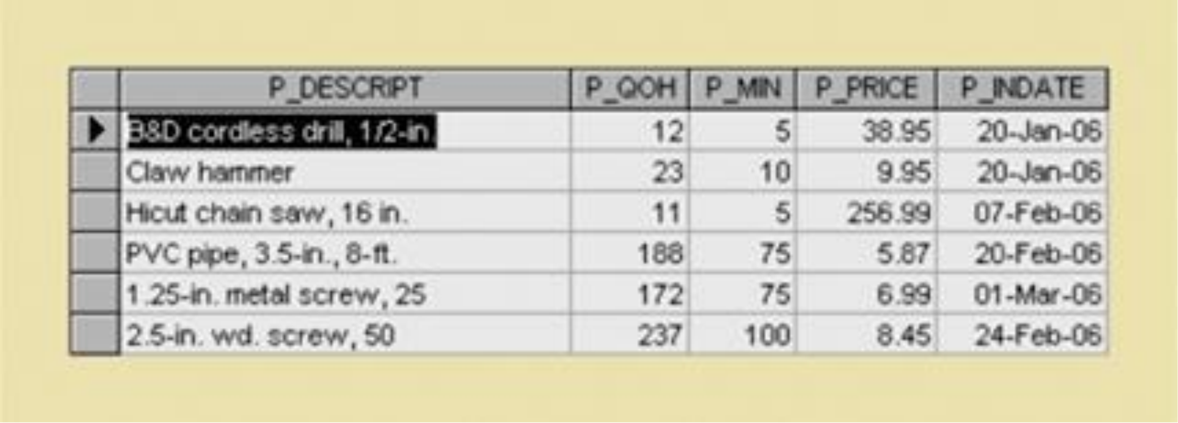


P_DESCRIPT	P_QOH	P_MIN	P_PRICE
Claw hammer	23	10	9.95
Rat-tail file, 1/8-in. fine	43	20	4.99
PVC pipe, 3.5-in., 8-ft	188	75	5.87
1.25-in. metal screw, 25	172	75	6.99
2.5-in. wd. screw, 50	237	100	8.45

Example 4

```
SELECT P_DESCRIP, P_QOH, P_MIN, P_PRICE, P_INDATE
FROM PRODUCT
WHERE P_INDATE >= '4-Jan-2006';
```

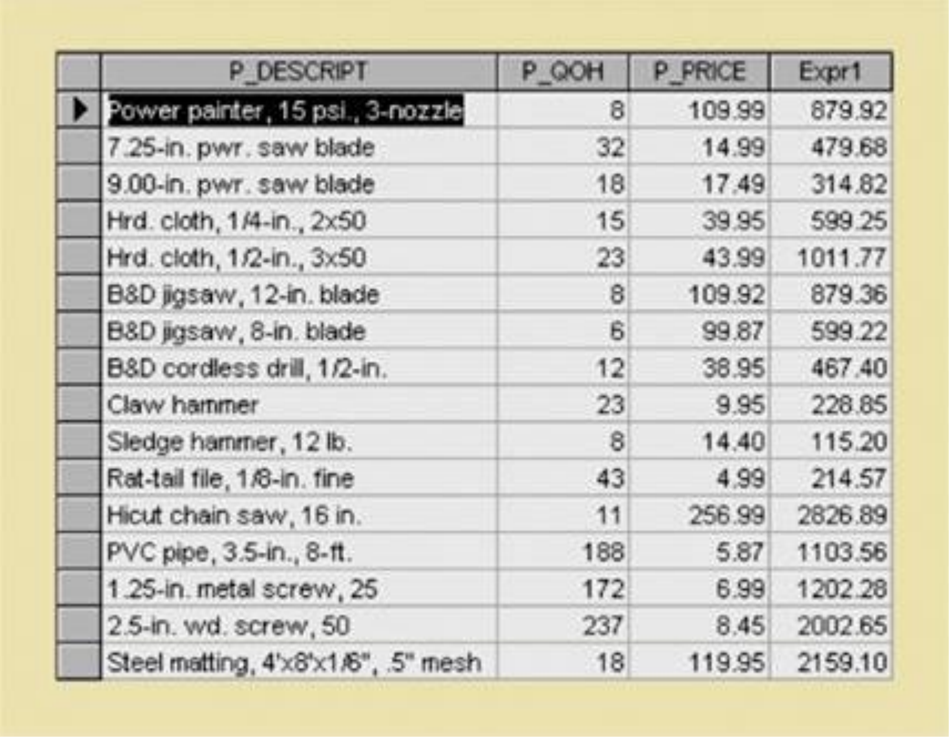
date comparison



	P_DESCRIPT	P_QOH	P_MIN	P_PRICE	P_INDATE
▶	B&D cordless drill, 1/2-in	12	5	38.95	20-Jan-06
	Claw hammer	23	10	9.95	20-Jan-06
	Hicut chain saw, 16 in.	11	5	256.99	07-Feb-06
	PVC pipe, 3.5-in., 8-ft.	188	75	5.87	20-Feb-06
	1.25-in. metal screw, 25	172	75	6.99	01-Mar-06
	2.5-in. wd. screw, 50	237	100	8.45	24-Feb-06

Example 5

```
SELECT P_DESCRIP, P_QOH, P_PRICE, P_QOH*P_PRICE
FROM PRODUCT;
```

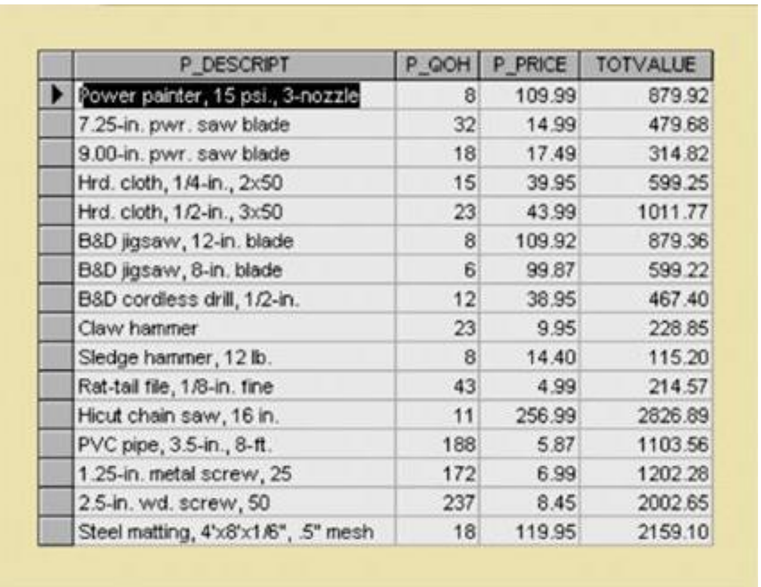


	P_DESCRIPT	P_QOH	P_PRICE	Expr1
▶	Power painter, 15 psi., 3-nozzle	8	109.99	879.92
	7.25-in. pwr. saw blade	32	14.99	479.68
	9.00-in. pwr. saw blade	18	17.49	314.82
	Hrd. cloth, 1/4-in., 2x50	15	39.95	599.25
	Hrd. cloth, 1/2-in., 3x50	23	43.99	1011.77
	B&D jigsaw, 12-in. blade	8	109.92	879.36
	B&D jigsaw, 8-in. blade	6	99.87	599.22
	B&D cordless drill, 1/2-in.	12	38.95	467.40
	Claw hammer	23	9.95	228.85
	Sledge hammer, 12 lb.	8	14.40	115.20
	Rat-tail file, 1/8-in. fine	43	4.99	214.57
	Hicut chain saw, 16 in.	11	256.99	2826.89
	PVC pipe, 3.5-in., 8-ft.	188	5.87	1103.56
	1.25-in. metal screw, 25	172	6.99	1202.28
	2.5-in. wd. screw, 50	237	8.45	2002.65
	Steel matting, 4'x8'x1/6", .5" mesh	18	119.95	2159.10

Example 6

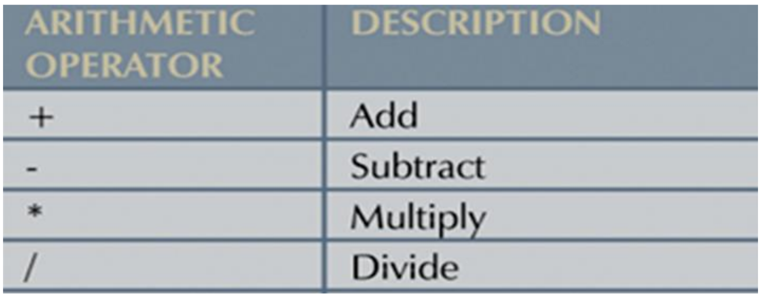
```
SELECT P_DESCRIP, P_QOH, P_PRICE, P_QOH*P_PRICE AS TOTVALUE
FROM PRODUCT;
```

computed column and an alias



P_DESCRIPT	P_QOH	P_PRICE	TOTVALUE
Power painter, 15 psi., 3-nozzle	8	109.99	879.92
7.25-in. pwr. saw blade	32	14.99	479.68
9.00-in. pwr. saw blade	18	17.49	314.82
Hrd. cloth, 1/4-in., 2x50	15	39.95	599.25
Hrd. cloth, 1/2-in., 3x50	23	43.99	1011.77
B&D jigsaw, 12-in. blade	8	109.92	879.36
B&D jigsaw, 8-in. blade	6	99.87	599.22
B&D cordless drill, 1/2-in.	12	38.95	467.40
Claw hammer	23	9.95	228.85
Sledge hammer, 12 lb.	8	14.40	115.20
Rat-tail file, 1/8-in. fine	43	4.99	214.57
Hicut chain saw, 16 in.	11	256.99	2826.89
PVC pipe, 3.5-in., 8-ft.	188	5.87	1103.56
1.25-in. metal screw, 25	172	6.99	1202.28
2.5-in. wd. screw, 50	237	8.45	2002.65
Steel matting, 4'x8'x1/6", .5" mesh	18	119.95	2159.10

Arithmetic Operators



ARITHMETIC OPERATOR	DESCRIPTION
+	Add
-	Subtract
*	Multiply
/	Divide

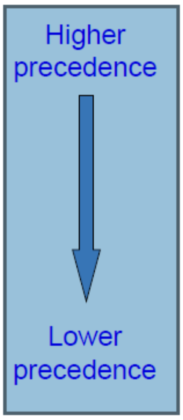
Precedence Rule

•Perform **parentheses/brackets**

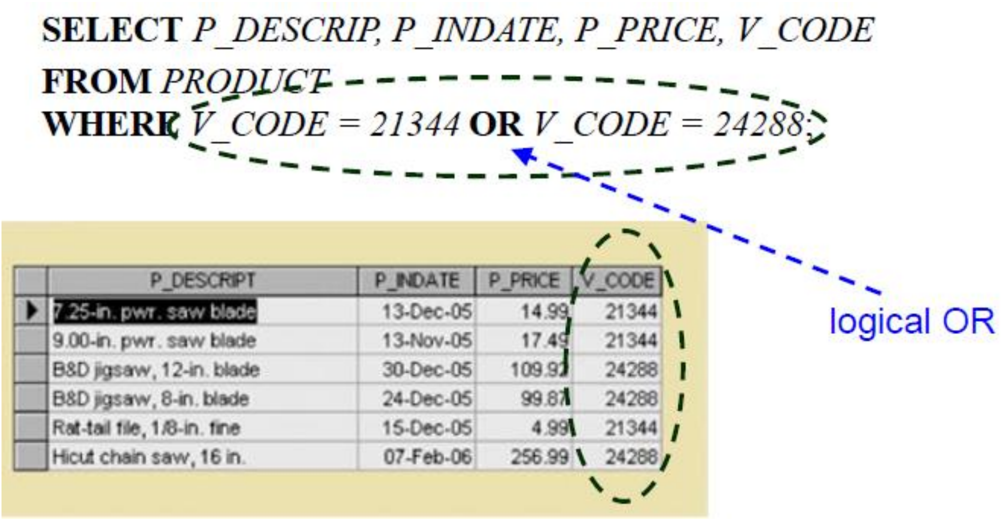
•Perform **power**

•Perform **multiplications and divisions**

•Perform **additions and subtractions**

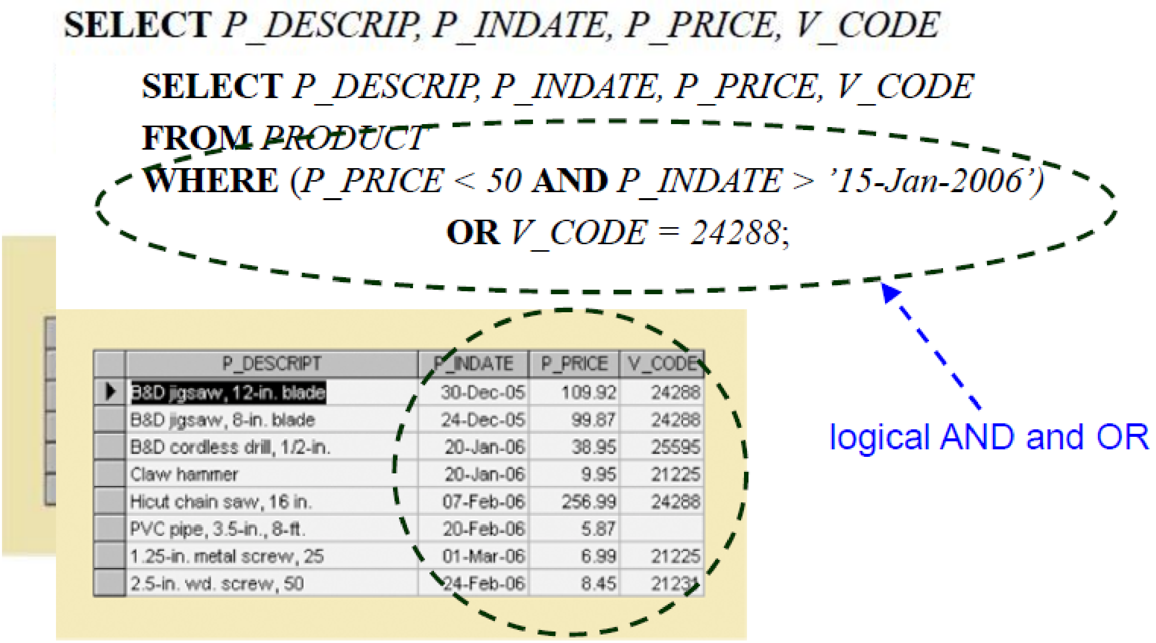


Logical Operators: AND, OR and NOT



```
SELECT P_DESCRIP, P_INDATE, P_PRICE, V_CODE
FROM PRODUCT
WHERE V_CODE = 21344 OR V_CODE = 24288;
```

	P_DESCRIPT	P_INDATE	P_PRICE	V_CODE
▶	7.25-in. pwr. saw blade	13-Dec-05	14.99	21344
	9.00-in. pwr. saw blade	13-Nov-05	17.49	21344
	B&D jigsaw, 12-in. blade	30-Dec-05	109.92	24288
	B&D jigsaw, 8-in. blade	24-Dec-05	99.87	24288
	Rat-tail file, 1/8-in. fine	15-Dec-05	4.99	21344
	Hicut chain saw, 16 in.	07-Feb-06	256.99	24288



```
SELECT P_DESCRIP, P_INDATE, P_PRICE, V_CODE
```

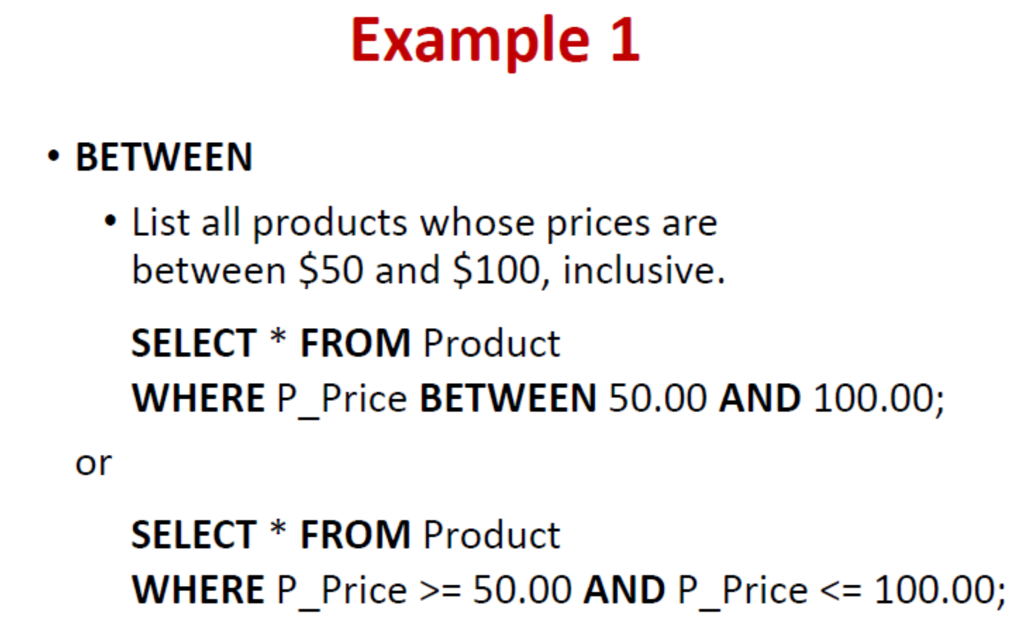
```
SELECT P_DESCRIP, P_INDATE, P_PRICE, V_CODE
FROM PRODUCT
WHERE (P_PRICE < 50 AND P_INDATE > '15-Jan-2006')
            OR V_CODE = 24288;
```

	P_DESCRIPT	P_INDATE	P_PRICE	V_CODE
▶	B&D jigsaw, 12-in. blade	30-Dec-05	109.92	24288
	B&D jigsaw, 8-in. blade	24-Dec-05	99.87	24288
	B&D cordless drill, 1/2-in.	20-Jan-06	38.95	25595
	Claw hammer	20-Jan-06	9.95	21225
	Hicut chain saw, 16 in.	07-Feb-06	256.99	24288
	PVC pipe, 3.5-in., 8-ft.	20-Feb-06	5.87	
	1.25-in. metal screw, 25	01-Mar-06	6.99	21225
	2.5-in. wd. screw, 50	24-Feb-06	8.45	21231

Special Operators

- **BETWEEN**
  - Used to check whether an attribute value is within a range, including the end points
- **IS NULL**
  - Used to check whether an attribute value is null
- **LIKE**
  - Used to check whether an attribute value matches a string pattern
- **IN**

  - Used to check whether an attribute value matches any value within a value list
- **EXISTS**
  - Used to check if a sub query returns any rows



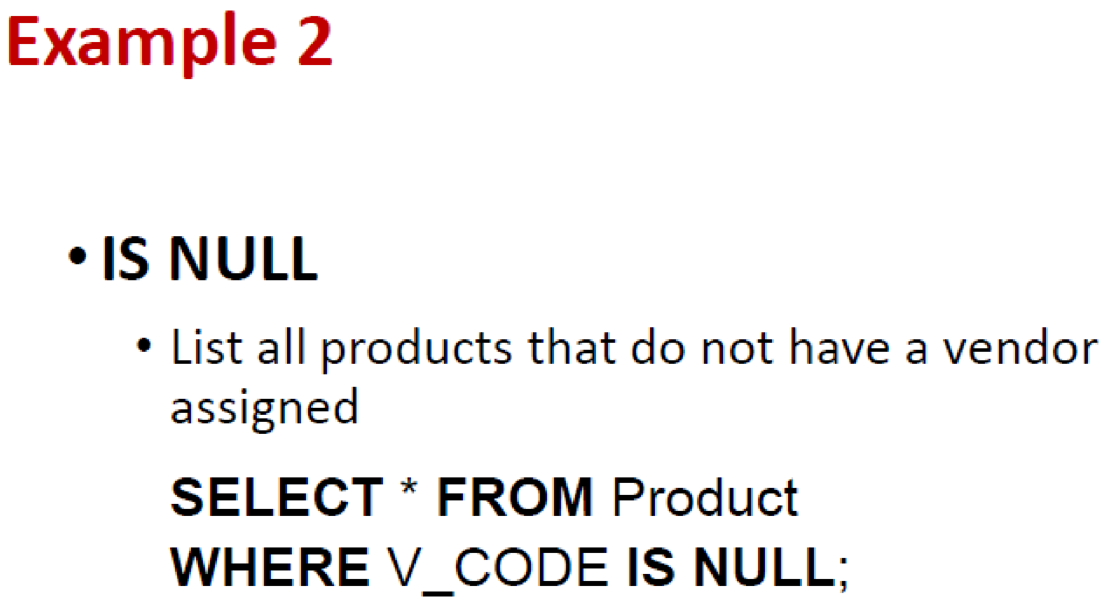
Example 1

- **BETWEEN**
  - List all products whose prices are between $50 and $100, inclusive.

```
SELECT * FROM Product
WHERE P_Price BETWEEN 50.00 AND 100.00;
```

or

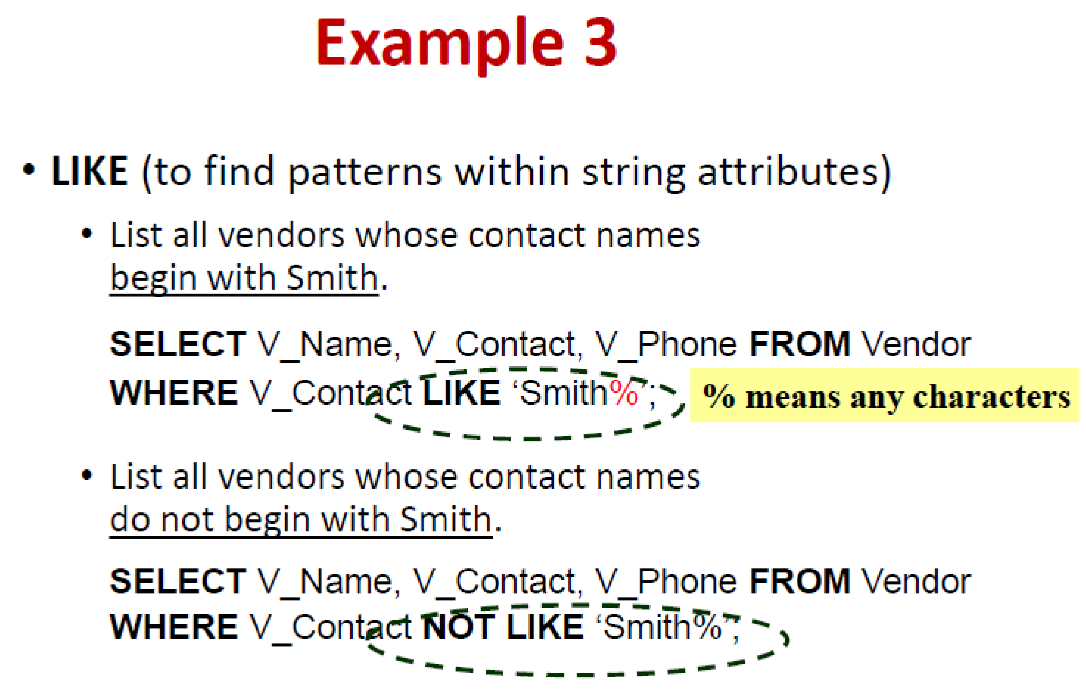
```
SELECT * FROM Product
WHERE P_Price >= 50.00 AND P_Price <= 100.00;
```



Example 2

- **IS NULL**
  - List all products that do not have a vendor assigned

```
SELECT * FROM Product
WHERE V_CODE IS NULL;
```



Example 3

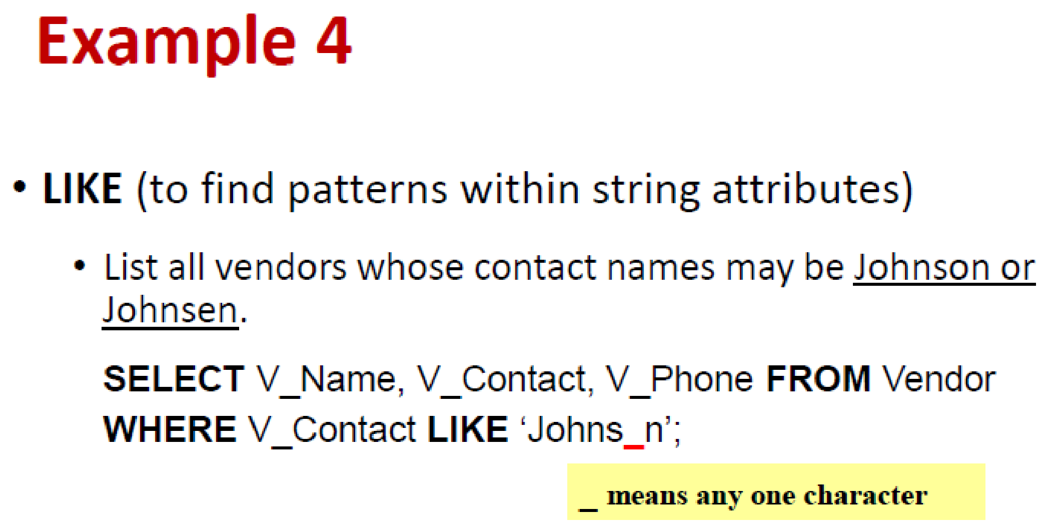
- **LIKE** (to find patterns within string attributes)
  - List all vendors whose contact names begin with Smith.

```
SELECT V_Name, V_Contact, V_Phone FROM Vendor
WHERE V_Contact LIKE 'Smith%';
```

% means any characters

  - List all vendors whose contact names do not begin with Smith.

```
SELECT V_Name, V_Contact, V_Phone FROM Vendor
WHERE V_Contact NOT LIKE 'Smith%';
```



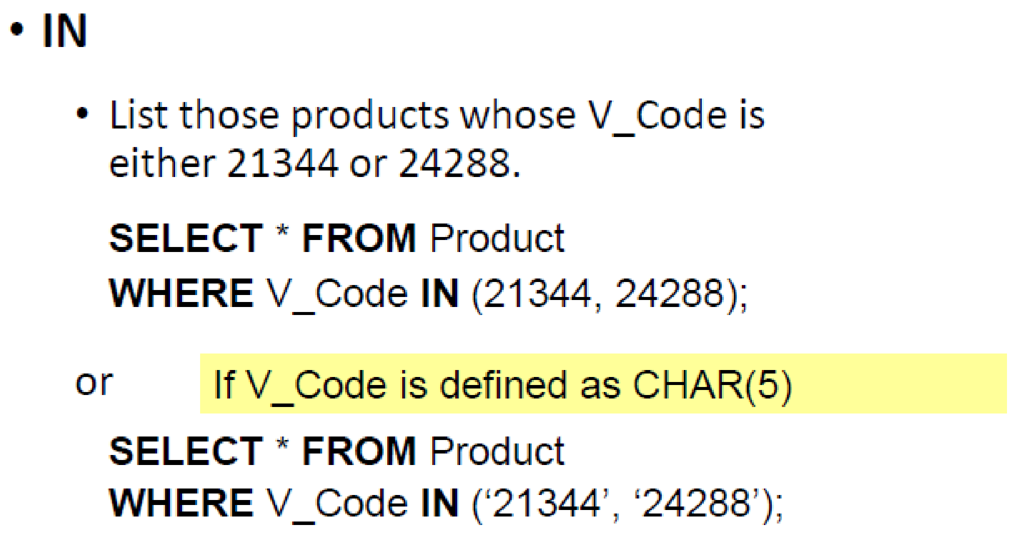
Example 4

- **LIKE** (to find patterns within string attributes)
  - List all vendors whose contact names may be Johnson or Johnsen.

```
SELECT V_Name, V_Contact, V_Phone FROM Vendor
WHERE V_Contact LIKE 'Johns_n';
```

_ means any one character

Example 5



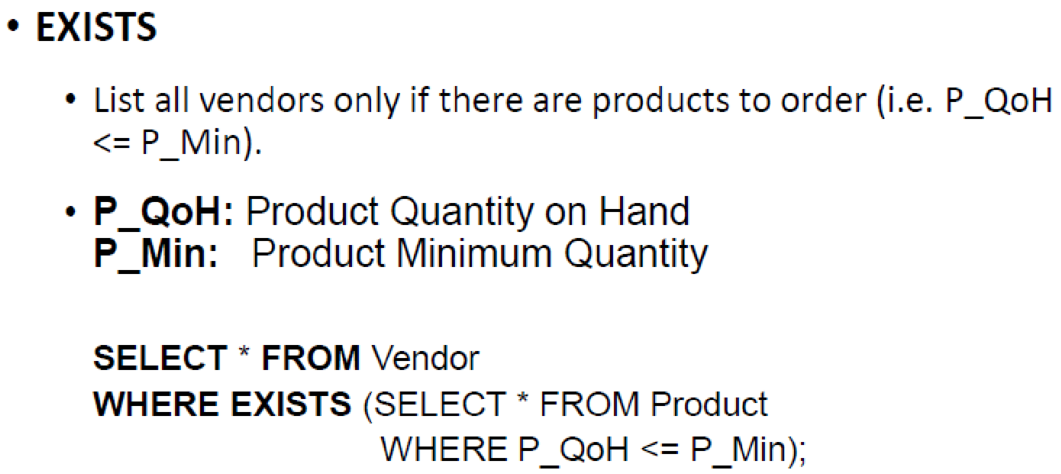
- **IN**
  - List those products whose V_Code is either 21344 or 24288.

```
SELECT * FROM Product
WHERE V_Code IN (21344, 24288);
```

or If V_Code is defined as CHAR(5)

```
SELECT * FROM Product
WHERE V_Code IN ('21344', '24288');
```

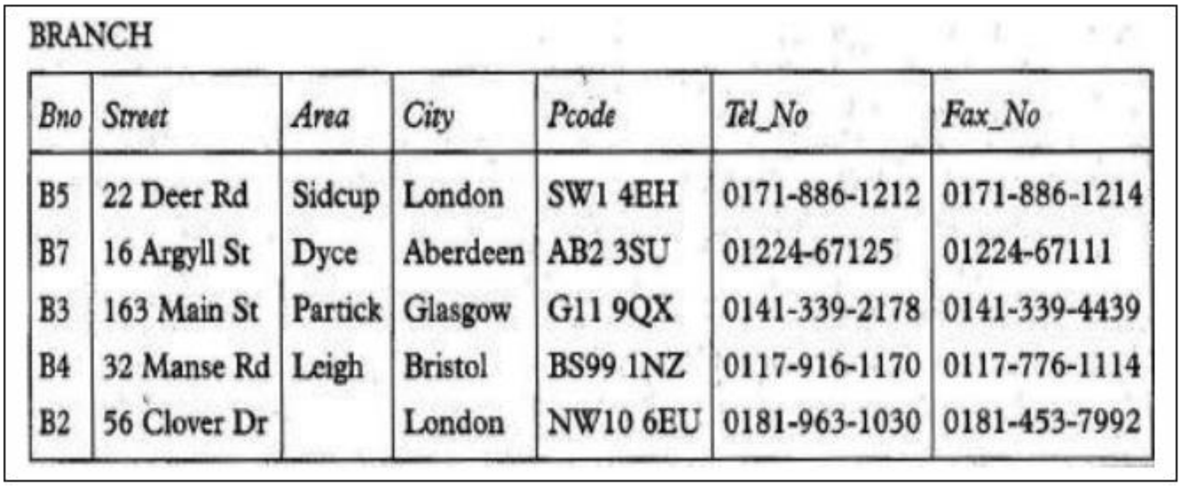
Example 6



- **EXISTS**
  - List all vendors only if there are products to order (i.e. P_QoH <= P_Min).
  - **P_QoH:** Product Quantity on Hand
    P_Min: Product Minimum Quantity

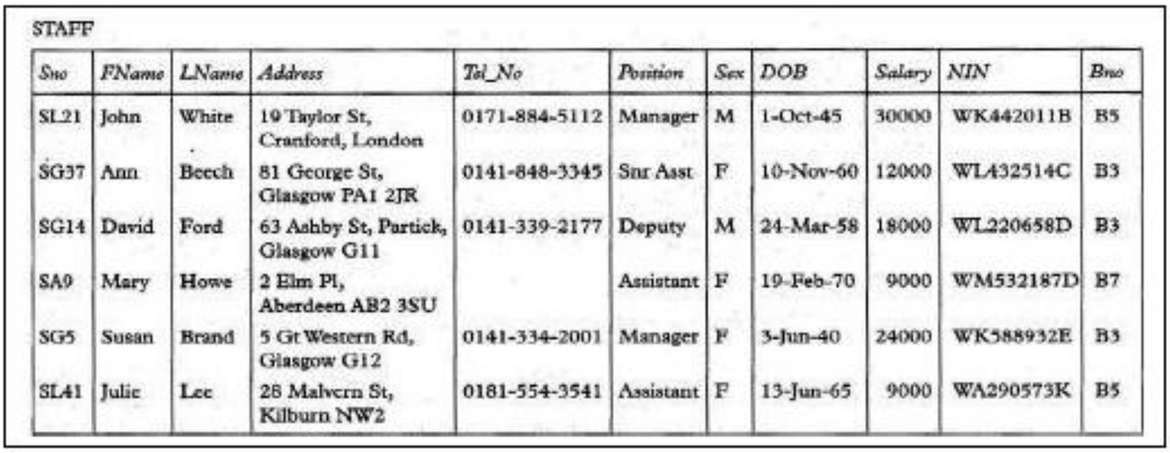
```
SELECT * FROM Vendor
WHERE EXISTS (SELECT * FROM Product
                        WHERE P_QoH <= P_Min);
```

Sample Database for Data Retrieval Using SELECT Statements



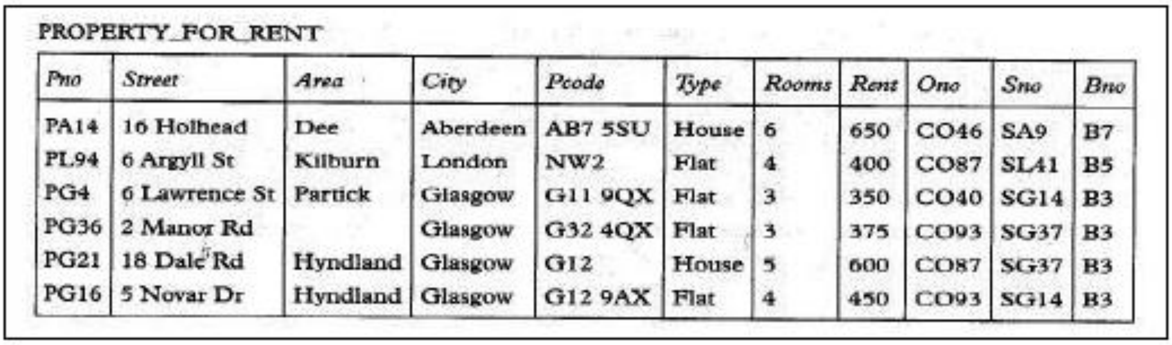
BRANCH

Bno	Street	Area	City	Pcode	Tel_No	Fax_No
B5	22 Deer Rd	Sidcup	London	SW1 4EH	0171-886-1212	0171-886-1214
B7	16 Argyll St	Dyce	Aberdeen	AB2 3SU	01224-67125	01224-67111
B3	163 Main St	Partick	Glasgow	G11 9QX	0141-339-2178	0141-339-4439
B4	32 Manse Rd	Leigh	Bristol	BS99 1NZ	0117-916-1170	0117-776-1114
B2	56 Clover Dr		London	NW10 6EU	0181-963-1030	0181-453-7992



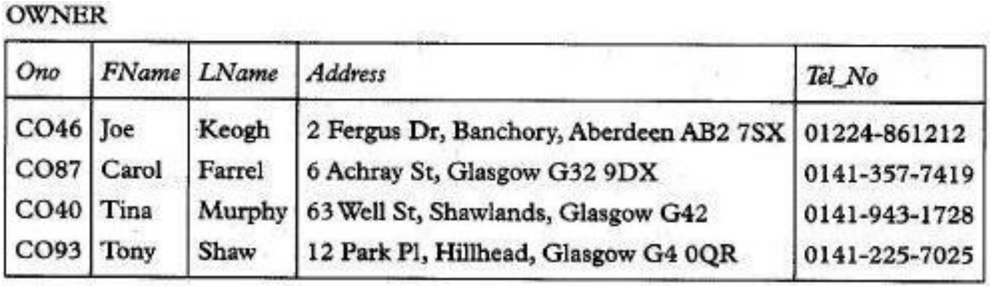
STAFF

Sno	FName	LName	Address	Tel_No	Position	Sex	DOB	Salary	NIN	Bno
SL21	John	White	19 Taylor St, Cranford, London	0171-884-5112	Manager	M	1-Oct-45	30000	WK442011B	B5
SG37	Ann	Beech	81 George St, Glasgow PA1 2JR	0141-848-3345	Snr Asst	F	10-Nov-60	12000	WL432514C	B3
SG14	David	Ford	63 Ashby St, Partick, Glasgow G11	0141-339-2177	Deputy	M	24-Mar-58	18000	WL220658D	B3
SA9	Mary	Howe	2 Elm Pl, Aberdeen AB2 3SU		Assistant	F	19-Feb-70	9000	WM532187D	B7
SG5	Susan	Brand	5 Gt Western Rd, Glasgow G12	0141-334-2001	Manager	F	3-Jun-40	24000	WK588932E	B3
SL41	Julie	Lee	28 Malvern St, Kilburn NW2	0181-554-3541	Assistant	F	13-Jun-65	9000	WA290573K	B5



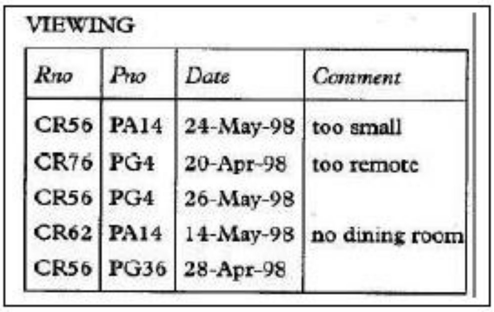
PROPERTY_FOR_RENT

Pno	Street	Area	City	Pcode	Type	Rooms	Rent	Ono	Sno	Bno
PA14	16 Holhead	Dee	Aberdeen	AB7 5SU	House	6	650	CO46	SA9	B7
PL94	6 Argyll St	Kilburn	London	NW2	Flat	4	400	CO87	SL41	B5
PG4	6 Lawrence St	Partick	Glasgow	G11 9QX	Flat	3	350	CO40	SG14	B3
PG36	2 Manor Rd		Glasgow	G32 4QX	Flat	3	375	CO93	SG37	B3
PG21	18 Dale Rd	Hyndland	Glasgow	G12	House	5	600	CO87	SG37	B3
PG16	5 Novar Dr	Hyndland	Glasgow	G12 9AX	Flat	4	450	CO93	SG14	B3



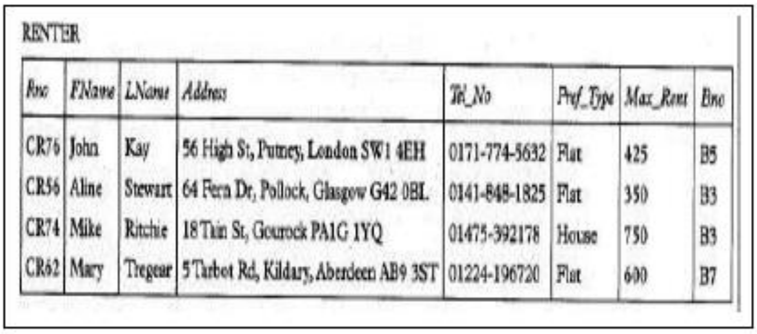
OWNER

Ono	FName	LName	Address	Tel_No
CO46	Joe	Keogh	2 Fergus Dr, Banchory, Aberdeen AB2 7SX	01224-861212
CO87	Carol	Farrel	6 Achray St, Glasgow G32 9DX	0141-357-7419
CO40	Tina	Murphy	63 Well St, Shawlands, Glasgow G42	0141-943-1728
CO93	Tony	Shaw	12 Park Pl, Hillhead, Glasgow G4 0QR	0141-225-7025



VIEWING

Rno	Pno	Date	Comment
CR56	PA14	24-May-98	too small
CR76	PG4	20-Apr-98	too remote
CR56	PG4	26-May-98	
CR62	PA14	14-May-98	no dining room
CR56	PG36	28-Apr-98	



RENTER

Rno	FName	LName	Address	Tel_No	Pref_Type	Max_Rent	Bno
CR76	John	Kay	56 High St, Putney, London SW1 4EH	0171-774-5632	Flat	425	B5
CR56	Aline	Stewart	64 Fern Dr, Pollock, Glasgow G42 0BL	0141-848-1825	Flat	350	B3
CR74	Mike	Ritchie	18 Tain St, Gourock PA1G 1YQ	01475-392178	House	750	B3
CR62	Mary	Tregear	5 Tarbot Rd, Kildary, Aberdeen AB9 3ST	01224-196720	Flat	600	B7

Example 1: Retrieving All Columns, All Rows

List full details of all staff.

Example 2: Retrieving Specific Columns in all rows

Produce a list of salaries for all staff, showing only the staff number, first and last names and the salary details

Example 3: Use of DISTINCT

List the property numbers of all properties that have been viewed

Example 4: Calculated Fields

Produce a list of monthly salaries for all staff, showing the staff number, first and last names and the salary details.

Example 5: Comparison Search Condition

List all staff with a salary greater than 10,000.

Example 6: Compound Comparison Search Condition

List the addresses of all branch offices in London or Glasgow.

Example 7: Range Search Condition

List all staff whose date of birth is between 01/01/1955 and 31/12/1975.

Example 8: Set Membership Search

List all Managers and Deputy Managers.

Example 9: Pattern Match Search Condition

Find all staff with the string 'Glasgow' in their address.

Example 10: NULL Search Condition (IS NULL/IS NOT NULL)

List the details of all viewings on property PG4 where a comment has not been supplied.

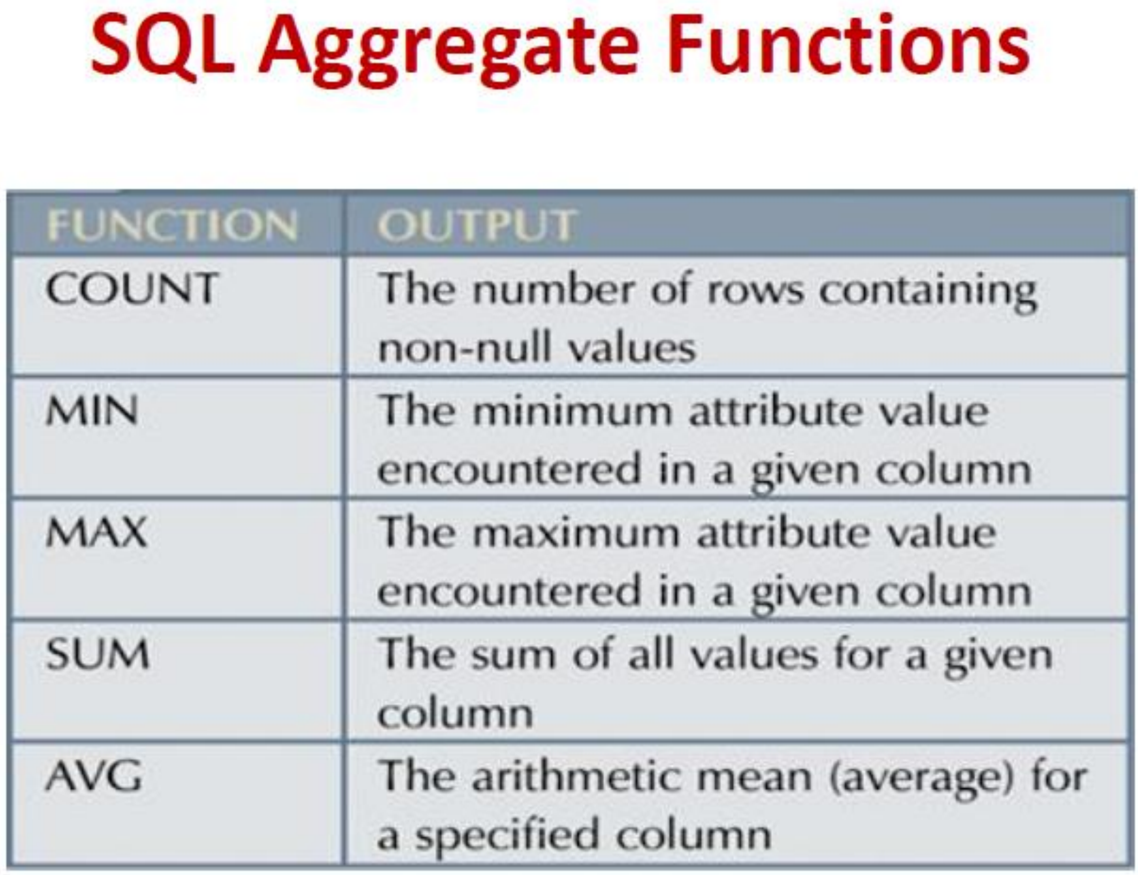
Sorting Results (ORDER BY Clause)

- Example : Single Column Ordering

  List salaries for all staff, arranged in descending order of salary.

- Example : Multiple Column Ordering

  Produce an abbreviated list of properties in order of property type and descending rent value within each type.



SQL Aggregate Functions

FUNCTION	OUTPUT
COUNT	The number of rows containing non-null values
MIN	The minimum attribute value encountered in a given column
MAX	The maximum attribute value encountered in a given column
SUM	The sum of all values for a given column
AVG	The arithmetic mean (average) for a specified column

Aggregate Functions

Question:

Can we use the following query to select products whose price equals the highest price of all the products?

```
SELECTP_CODE, P_DESCRIPT, P_PRICE

FROMPRODUCT

WHEREP_PRICE = MAX(P_PRICE);
```

- **Example : Use of COUNT(*)**

  How many properties cost more than £350 per month to rent?

- **Example : Use of COUNT(DISTINCT)**

  How many different properties were viewed in May 1995?

- **Example : Use of COUNT and SUM**

  Find total number of Managers and the sum of their salaries.

- **Example : Use of MIN, MAX, AVG**

  Find the minimum, maximum and average staff salary.

Grouping Results (GROUP BY Clause)

- Can use GROUP BY clause of SELECT statement to get sub-totals.
- SELECT and GROUP BY clauses must be 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.
- If WHERE is used with GROUP BY, WHERE is applied first, then groups are formed from remaining rows satisfying the predicate
- **Example : Use of GROUP BY**

  Find the number of staff in each branch and their total salaries.

Joining Database Tables

- Joining tables on common attributes is an important distinction in relational database
- Join is performed when retrieving data from more than one table at a time
- Join is generally composed of an equality comparison between foreign key and primary key of related tables
- This general idea is:

  get rows from Table A

  and rows from Table B

  where the PK(Table A) = FK(Table B)

Multi-table Queries

• Example : Simple Join

- List names of all renters who have viewed a property along with any comment supplied.

```
SELECT r.rno, fname, lname, pno, comment
FROM renter r, viewing v
WHERE r.rno = v.rno;
OR
SELECT renter.rno, fname, lname, pno, comment
FROM renter, viewing
WHERE renter.rno = viewing.rno;
```

- To obtain correct rows, include only those rows from both tables that have identical values in the **rno** column: r.rno = v.rno.
- These two columns are the matching columns for two tables.
- This is equivalent to the equi-join in relational algebra

• Example : Sorting a Join

- For each branch, list names of staff who manage properties.

```
SELECT s.bno, s.sno, fname, lname, pno
FROM staff s, property_for_rent p
WHERE s.sno = p.sno
ORDER BY s.bno, s.sno, pno;
```

• Example : Three Table Join

- For each branch, list staff who manage properties, including the city in which the branch is located and properties they manage.

```
SELECT b.bno, b.city, s.sno, fname, lname, pno
FROM branch b, staff s, property_for_rent p
WHERE b.bno = s.bno AND s.sno = p.sno
ORDER BY b.bno, s.sno, pno;
OR
SELECT branch.bno, branch.city, staff.sno, fname, lname, pno
FROM branch, staff, property_for_rent
WHERE branch.bno = staff.bno AND staff.sno = property_for_rent.sno
ORDER BY branch.bno, staff.sno, pno;
```

• Example : Multiple Grouping Columns

- Find number of properties handled by each staff member in each branch.

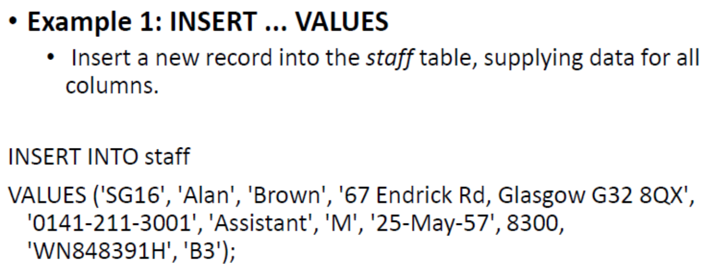
```
SELECT bno, sno, COUNT(*) AS count
FROM staff s , property_for-rent p
WHERE s.sno = p.sno
GROUP BY s.bno, s.sno
ORDER BY s.bno, s.sno;
```

Data Manipulation (Database Updates)

- **Insert**

```
INSERT INTO table_name[ (column_list) ]

VALUES (data_value_list)
```

- *Column_list*is optional.
- If omitted, SQL assumes a list of all columns in their original CREATE TABLE order.
- Any columns omitted must have been declared as NULL when table was created, unless DEFAULT was specified when creating column.
- *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 both lists.
- Data type of each item in data_value_listmust be compatible with data type of corresponding column.

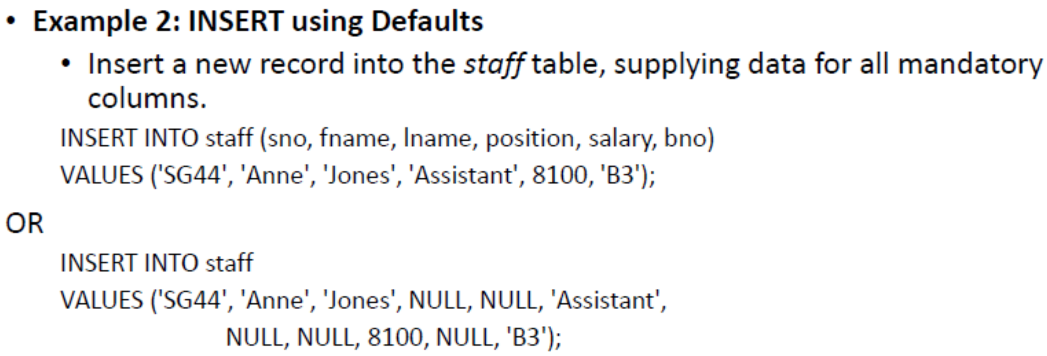


- **Example 1: INSERT ... VALUES**
  - Insert a new record into the *staff* table, supplying data for all columns.

```
INSERT INTO staff
VALUES ('SG16', 'Alan', 'Brown', '67 Endrick Rd, Glasgow G32 8QX',
   '0141-211-3001', 'Assistant', 'M', '25-May-57', 8300,
   'WN848391H', 'B3');
```

Adding Table Rows

- When entering values, notice that:
  - Row contents are entered between **parentheses**
  - Character and date values are entered between **apostrophes**
  - Numerical entries are **not enclosed in apostrophes**
  - Attribute entries are separated by **commas**
  - A value is required for **each column**
  - Use NULL for unknown values

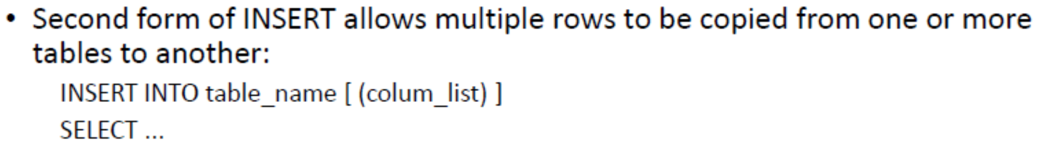


- **Example 2: INSERT using Defaults**
  - Insert a new record into the *staff* table, supplying data for all mandatory columns.

```
INSERT INTO staff (sno, fname, lname, position, salary, bno)
VALUES ('SG44', 'Anne', 'Jones', 'Assistant', 8100, 'B3');
```

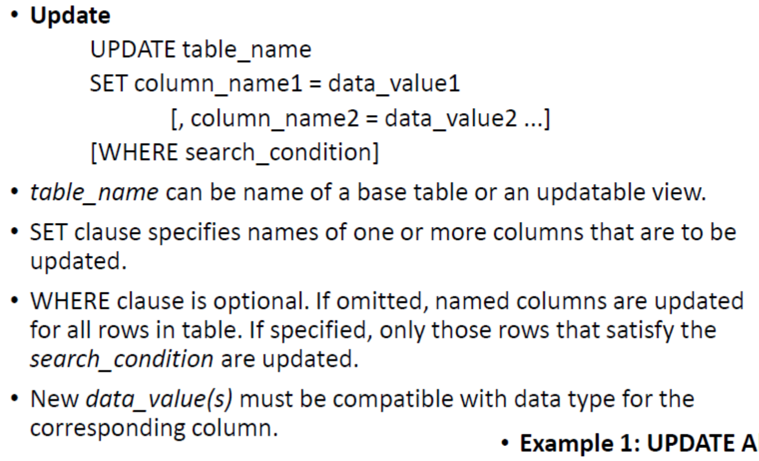
OR

```
INSERT INTO staff
VALUES ('SG44', 'Anne', 'Jones', NULL, NULL, 'Assistant',
                NULL, NULL, 8100, NULL, 'B3');
```



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

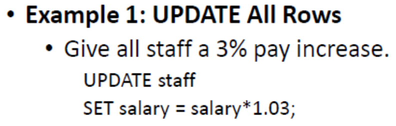
```
INSERT INTO table_name [ (colum_list) ]
SELECT ...
```



- **Update**

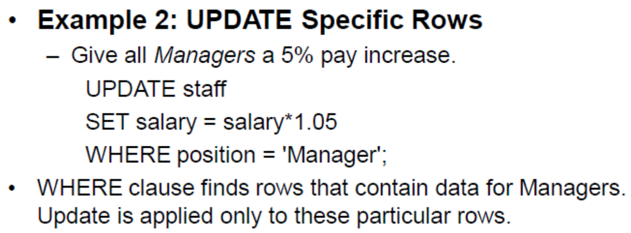
```
UPDATE table_name
SET column_name1 = data_value1
        [, column_name2 = data_value2 ...]
[WHERE search_condition]
```

- *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.
- WHERE clause is optional. If omitted, named columns are updated for all rows in table. If specified, only those rows that satisfy the *search_condition* are updated.
- New *data_value(s)* must be compatible with data type for the corresponding column.



- **Example 1: UPDATE All Rows**
  - Give all staff a 3% pay increase.

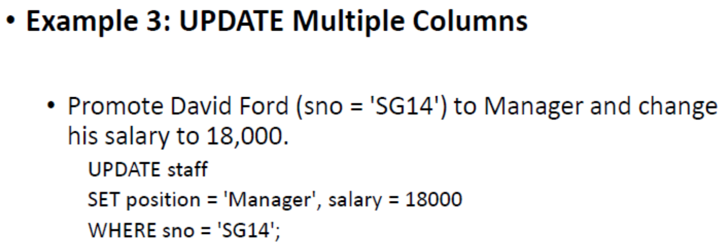
```
UPDATE staff
SET salary = salary*1.03;
```



- **Example 2: UPDATE Specific Rows**
  – Give all *Managers* a 5% pay increase.

```
UPDATE staff
SET salary = salary*1.05
WHERE position = 'Manager';
```

- WHERE clause finds rows that contain data for Managers. Update is applied only to these particular rows.



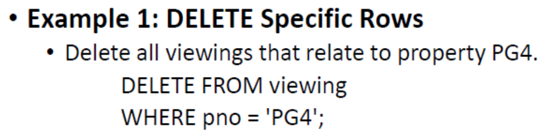
- **Example 3: UPDATE Multiple Columns**
  - Promote David Ford (sno = 'SG14') to Manager and change his salary to 18,000.

```
UPDATE staff
SET position = 'Manager', salary = 18000
WHERE sno = 'SG14';
```

•Delete

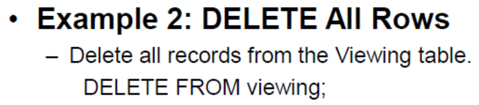
```
DELETE FROM table_name

[WHERE search_condition]
```

- *table_name*can be name of a base table or an updatable view.
- *Search condition* is optional; if omitted, all rows are deleted from the table.
- This does not delete the table. If *search_condition*is specified, only those rows that satisfy the condition are deleted.



- **Example 1: DELETE Specific Rows**
  - Delete all viewings that relate to property PG4.

```
DELETE FROM viewing
WHERE pno = 'PG4';
```



- **Example 2: DELETE All Rows**
  - – Delete all records from the Viewing table.

```
DELETE FROM viewing;
```

Data Definition (Table Creation)

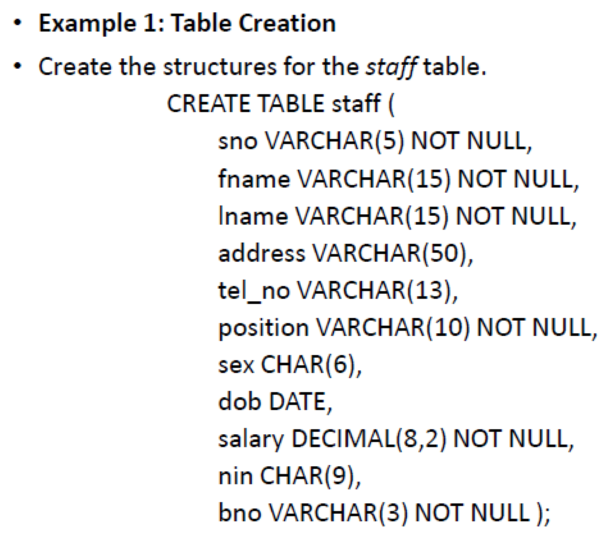
```
CREATE TABLE table_name

(column_namedata_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.

Common MySQL Data Types:

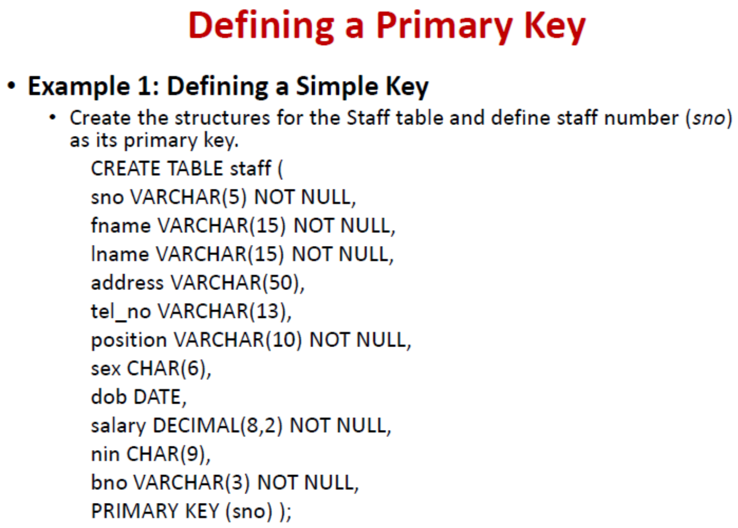
- CHAR(*size*) -Fixed length character data of length *size*
- DECIMAL(p, s)-DEC -Number having precision *p* and scale *s*.
- INTEGER()-INT –a basic whole number
- DATE -Valid data range

- VARCHAR(*size*) -Variable length character string having maximum length *size* bytes.



- **Example 1: Table Creation**
- Create the structures for the *staff* table.

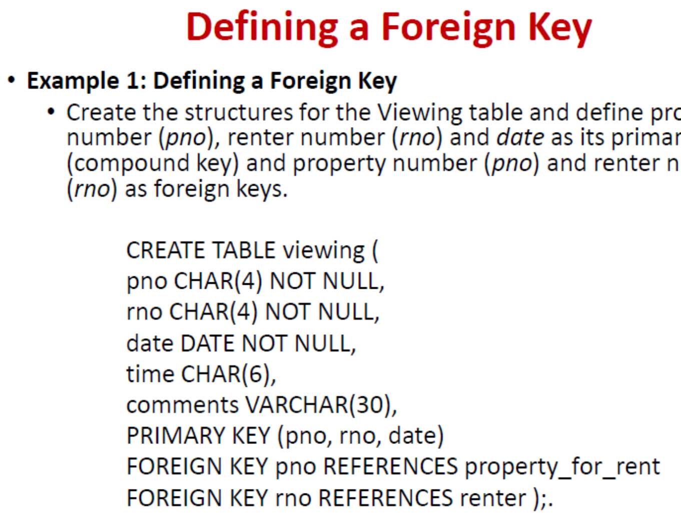
```
CREATE TABLE staff (
        sno VARCHAR(5) NOT NULL,
        fname VARCHAR(15) NOT NULL,
        lname VARCHAR(15) NOT NULL,
        address VARCHAR(50),
        tel_no VARCHAR(13),
        position VARCHAR(10) NOT NULL,
        sex CHAR(6),
        dob DATE,
        salary DECIMAL(8,2) NOT NULL,
        nin CHAR(9),
        bno VARCHAR(3) NOT NULL );
```



Defining a Primary Key

- **Example 1: Defining a Simple Key**
  - Create the structures for the Staff table and define staff number (*sno*) as its primary key.

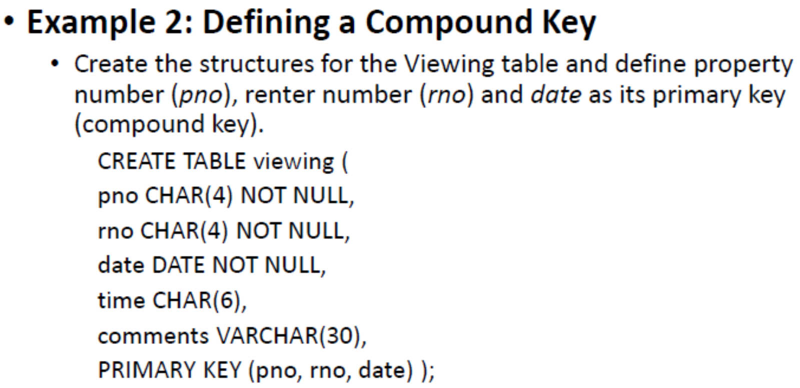
```
CREATE TABLE staff (
sno VARCHAR(5) NOT NULL,
fname VARCHAR(15) NOT NULL,
lname VARCHAR(15) NOT NULL,
address VARCHAR(50),
tel_no VARCHAR(13),
position VARCHAR(10) NOT NULL,
sex CHAR(6),
dob DATE,
salary DECIMAL(8,2) NOT NULL,
nin CHAR(9),
bno VARCHAR(3) NOT NULL,
PRIMARY KEY (sno) );
```



Defining a Foreign Key

- **Example 1: Defining a Foreign Key**
  - Create the structures for the Viewing table and define property number (*pno*), renter number (*rno*) and *date* as its primary key (compound key) and property number (*pno*) and renter number (*rno*) as foreign keys.

```
CREATE TABLE viewing (
pno CHAR(4) NOT NULL,
rno CHAR(4) NOT NULL,
date DATE NOT NULL,
time CHAR(6),
comments VARCHAR(30),
PRIMARY KEY (pno, rno, date)
FOREIGN KEY pno REFERENCES property_for_rent
FOREIGN KEY rno REFERENCES renter );.
```



- **Example 2: Defining a Compound Key**
  - Create the structures for the Viewing table and define property number (*pno*), renter number (*rno*) and *date* as its primary key (compound key).

```
CREATE TABLE viewing (
pno CHAR(4) NOT NULL,
rno CHAR(4) NOT NULL,
date DATE NOT NULL,
time CHAR(6),
comments VARCHAR(30),
PRIMARY KEY (pno, rno, date) );
```

Changing the Table Definition

- A table definition can be changed by:
- using the ALTER TABLE command, and
- a keyword for a specific change such as

ADD, **MODIFY,** **DELETE**

Modification of Table Structures

- **Example : Adding an Attribute**
  - Add an attribute called *nationality* to the Staff table.
- **Example : Adding a Primary Key Constraint**
  - Define *bno*(branch number) as the primary key of an existing table called *branch.*

Removing a Table (DDL)

- **DROP TABLE table_name**
  - Removes not only the named table but also all the rows within it.
- **Example 1: DELETE a Table**
  - **Remove the table *property_for_rent.***

Miscellaneous Table Commands

- **DESCRIBE *table-name***
  - Shows the details of the table structure you created
  - Good for verifying what you have done
- **TRUNCATE TABLE *table-name***
  - Remove all the records from the table

Manipulating Data

- Once a database and tables are created
  - DML commands can be written to delete, insert, edit
  - Queries can be written to retrieve data
- Commands
  - Adding table rows
  - Saving table changes (**COMMIT command**)
  - Restoring table contents (**ROLLBACK command**)
  - Listing table rows
  - Updating table rows
  - Deleting table rows
  - Inserting table rows with a select subquery

Reference

Lecture notes - Dr. Dilani Wickramaarachchi (University of Kelaniya)

https://www.w3schools.com/sql/sql_intro.asp