# Database Management Systems and SQL

## Lecture 1

## What is a DBMS?

- Collection of interrelated data – manual or computerized or online
- Set of programs to access the data
- DBMS provides an environment that is both ***convenient* and *efficient* to use.**

## Applications Areas of DBMS?

- ✓ Banking: all transactions
- ✓ Airlines: reservations, schedules
- ✓ Universities:  registration, grades
- ✓ Sales: customers, products, purchases
- ✓ Manufacturing: production, inventory, orders, supply chain
- ✓ Human resources:  employee records, salaries, tax deductions

# Why do we use DBMS

- To avoid data redundancy and inconsistency
  - ✓Multiple file formats, duplication of information in different files
- To avoid difficulty in accessing data
  - ✓Need to write a new program to carry out each new task
- To deal with data isolation — multiple files and formats
- To deal with integrity problems
  - ✓Integrity constraints (e.g. account balance > 0) become part of program code
  - ✓Easy to add new constraints or change existing ones

# Why do we use DBMS (contd..)

1. Atomicity of updates
   - Failures may leave database in an inconsistent state with partial updates carried out
   - E.g. transfer of funds from one account to another should either complete or not happen at all
2. Concurrent access by multiple users
   - Concurrent accessed needed for performance
   - Uncontrolled concurrent accesses can lead to inconsistencies
     - E.g. two people reading a balance and updating it at the same time
3. Security problems

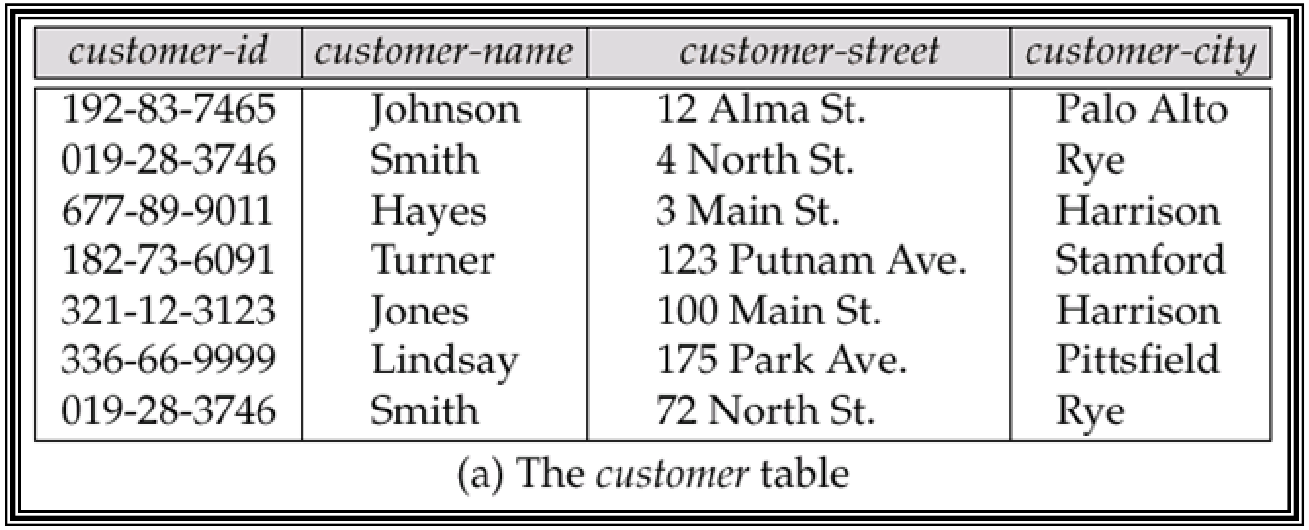
# Relational Model

Attribute
s

Example of tabular data in the relational model

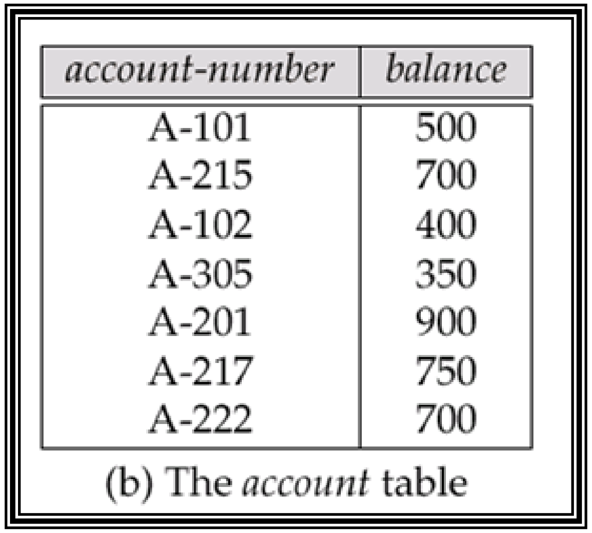
| customer-id | customer-name | customer-street | customer-city | account-number |
|---|---|---|---|---|
| 192-83-7465 | Johnson | Alma | Palo Alto | A-101 |
| 019-28-3746 | Smith | North | Rye | A-215 |
| 192-83-7465 | Johnson | Alma | Palo Alto | A-201 |
| 321-12-3123 | Jones | Main | Harrison | A-217 |
| 019-28-3746 | Smith | North | Rye | A-201 |

# A Logically Related Database



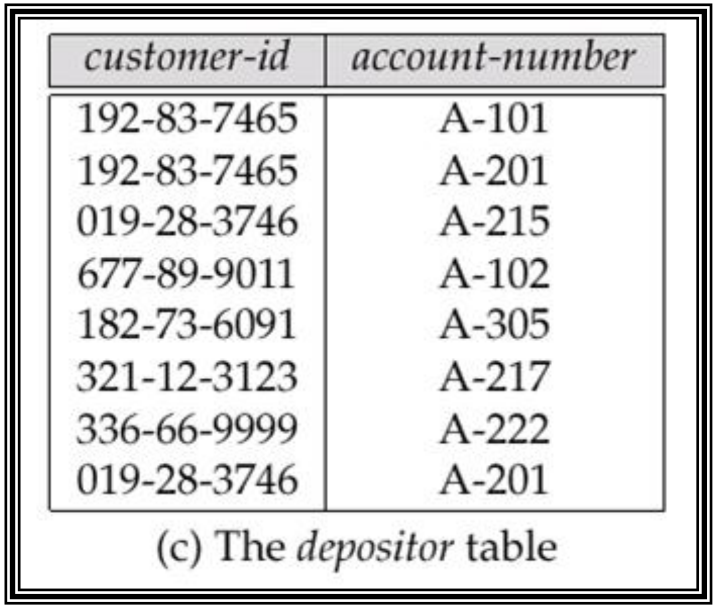
| *customer-id* | *customer-name* | *customer-street* | *customer-city* |
|---|---|---|---|
| 192-83-7465 | Johnson | 12 Alma St. | Palo Alto |
| 019-28-3746 | Smith | 4 North St. | Rye |
| 677-89-9011 | Hayes | 3 Main St. | Harrison |
| 182-73-6091 | Turner | 123 Putnam Ave. | Stamford |
| 321-12-3123 | Jones | 100 Main St. | Harrison |
| 336-66-9999 | Lindsay | 175 Park Ave. | Pittsfield |
| 019-28-3746 | Smith | 72 North St. | Rye |

(a) The *customer* table



| *account-number* | *balance* |
|---|---|
| A-101 | 500 |
| A-215 | 700 |
| A-102 | 400 |
| A-305 | 350 |
| A-201 | 900 |
| A-217 | 750 |
| A-222 | 700 |

(b) The *account* table



| *customer-id* | *account-number* |
|---|---|
| 192-83-7465 | A-101 |
| 192-83-7465 | A-201 |
| 019-28-3746 | A-215 |
| 677-89-9011 | A-102 |
| 182-73-6091 | A-305 |
| 321-12-3123 | A-217 |
| 336-66-9999 | A-222 |
| 019-28-3746 | A-201 |

(c) The *depositor* table

## Database Users

Users are differentiated by the way they expect to interact with the system DML (**Data Manipulation Language**)calls

- Sophisticated users – form requests in a database query language
- Specialized users – write specialized database applications that do not fit into the traditional data processing framework
- Naïve users – invoke one of the permanent application programs that have been written previously
  - ✓ E.g. people accessing database over the web, bank tellers, clerical staff

## Database Administrator

- Coordinates all the activities of the database system
- Has a good understanding of the enterprise's information resources and needs.
- Database administrator's responsibilities include:
  - ✓ Schema definition
  - ✓ Storage structure and access method definition
  - ✓ Schema and physical organization modification
  - ✓ Granting user authority to access the database
  - ✓ Specifying integrity constraints
  - ✓ Acting as liaison with users
  - ✓ Monitoring performance and responding to changes in requirements

## Transaction Management

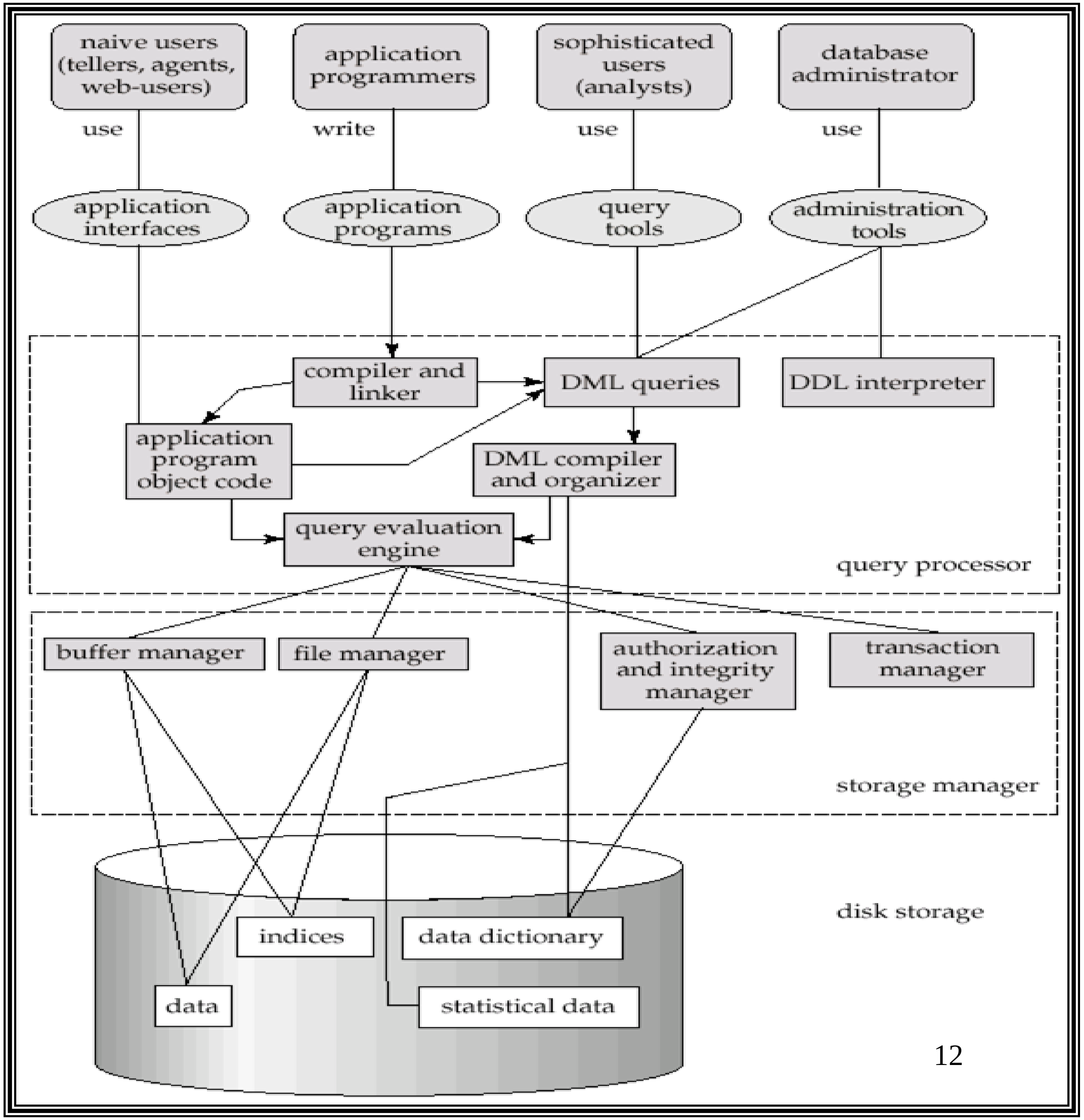
- **A *transaction* is a collection of operations that performs a single logical function in a database application**
- Transaction-management component ensures that the database remains in a consistent (correct) state despite system failures (**e.g., power failures and operating system crashes) and transaction failures)**.
- Concurrency-control manager controls the interaction among the concurrent transactions, to ensure the consistency of the database.

## Storage Management

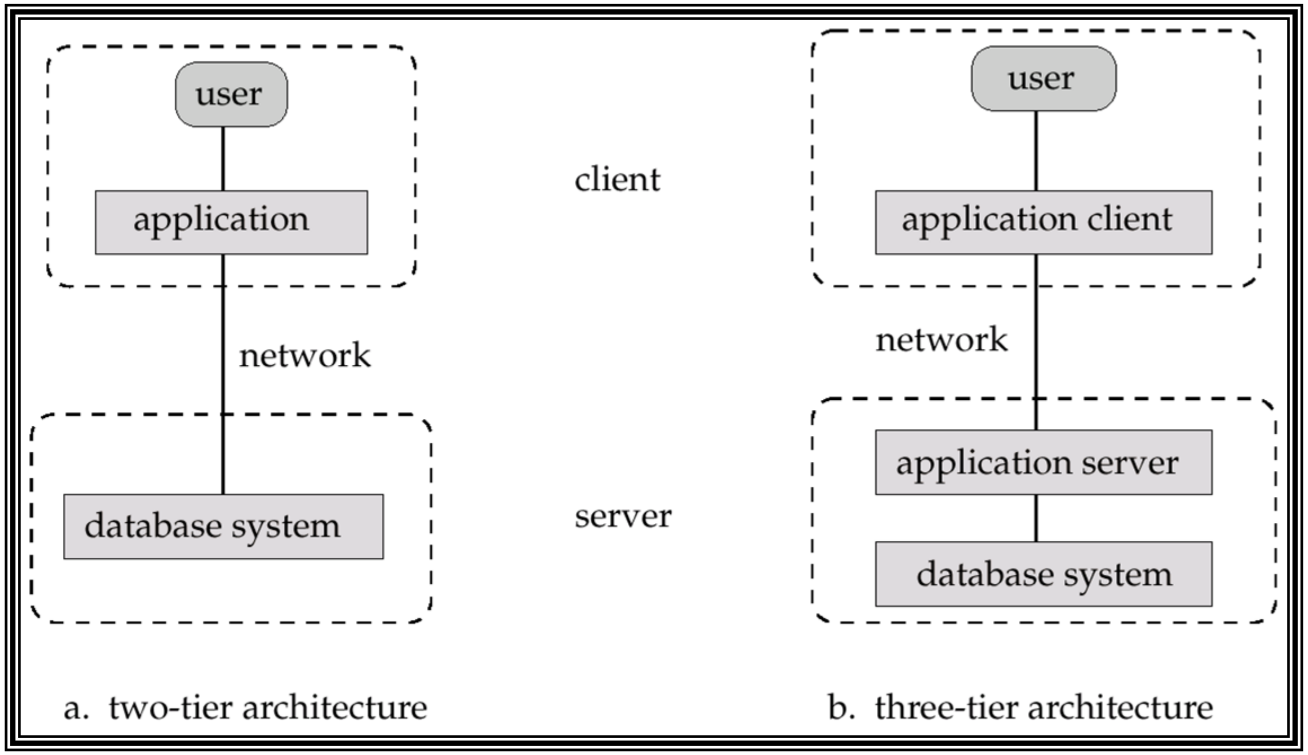
- Storage manager is a program module that provides the interface between the **low-level data stored in the database** and the **application programs** and **queries** submitted to the system.

- The storage manager is responsible to the following tasks:
  - interaction with the file manager
  - efficient storing, retrieving and updating of data

# Overall System Structure

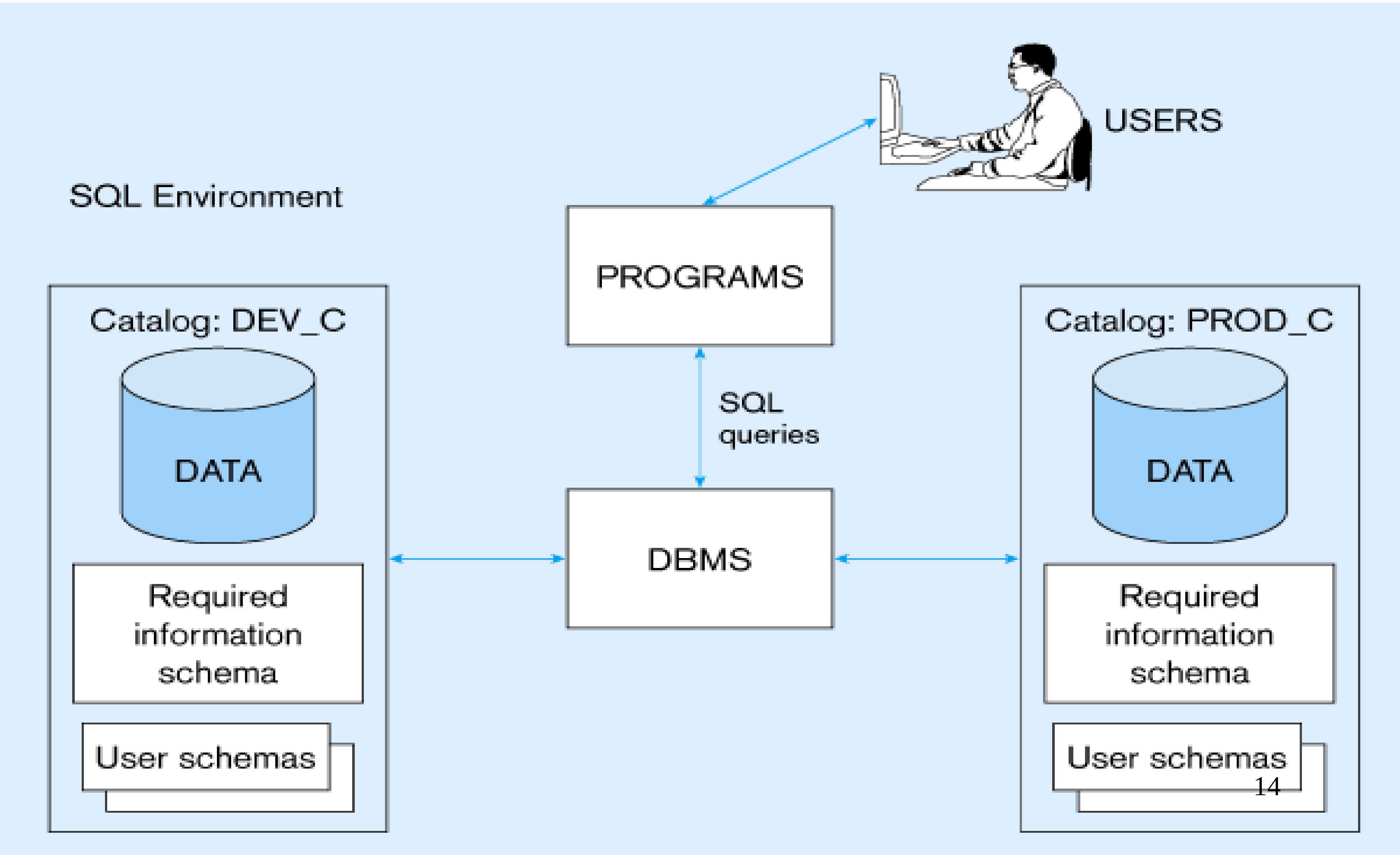


# Application Architectures



- **Two-tier architecture**: E.g. client programs using ODBC/JDBC to communicate with a database
- **Three-tier architecture**: E.g. web-based applications, and applications built using "middleware"

# DBMS: Allows to Create, Manipulate & Access the Data



## The Language of DBMS

# SQL

Structured Query Language

Standard language for **querying** and **manipulating** data. **Very widely used.**

1. Data Definition Language (DDL)
   – Create/alter/delete tables and their attributes
2. Data Manipulation Language (DML)
   – Insert/delete/modify tuples in tables

# SQL

- SQL: widely used non-procedural language
  - E.g. find the name of the customer with customer-id 192-83-7465

    **select** *customer.customer-name*
    **from** *customer*
    **where** *customer.customer-id* = ‘192-83-7465’

  - E.g. find the balances of all accounts held by the customer with customer-id 192-83-7465

    **select** *account.balance*
    **from** *depositor*, *account*
    **where** *depositor.customer-id* = ‘192-83-7465’ **and**
    *depositor.account-number* = *account.account-number*

- Application programs generally access databases through one of
  - Language extensions to allow embedded SQL
  - Application program interface (e.g. ODBC/JDBC) which allow SQL queries to be sent to a database

# Tables in RDBMS

Table name

Attribute names

Product

| PName | Price | Category | Manufacturer |
|---|---|---|---|
| Gizmo | 19.99 | Gadgets | GizmoWorks |
| Powergizmo | 29.99 | Gadgets | GizmoWorks |
| SingleTouch | 149.99 | Photography | Canon |
| MultiTouch | 203.99 | Household | Hitachi |

Tuples or rows

## Steps to Define the Schema

**Step 1: Define table name and its attributes**

Product(PName, Price, Category, Manufacturer)

Product

| PName | Price | Category | Manufacturer |
|---|---|---|---|
| Gizmo | 19.99 | Gadgets | GizmoWorks |
| Powergizmo | 29.99 | Gadgets | GizmoWorks |
| SingleTouch | 149.99 | Photography | Canon |
| MultiTouch | 203.99 | Household | Hitachi |

# Data Types and Domain of Attributes

Product(<u>PName</u>, Price, Category, Manfacturer)

Basic **data types**

- **Numeric**
  - Integer numbers: INTEGER, INT, and SMALLINT
  - Floating-point (real) numbers: FLOAT or REAL, and DOUBLE PRECISION
- **Character-string**
  - Fixed length: CHAR(*n*), CHARACTER(*n*)
  - Varying length: VARCHAR(*n*), CHAR VARYING(*n*), CHARACTER VARYING(*n*)

## Data Types and Domain of Attributes

- **Boolean**
  - Values of TRUE or FALSE or NULL
- **DATE**
  - Ten positions
  - Components are YEAR, MONTH, and DAY in the form YYYY-MM-DD
- **Timestamp**
  - Includes the DATE and TIME fields
  - Plus a minimum of six positions for decimal fractions of seconds
  - Optional WITH TIME ZONE qualifier

## Steps to Define the Schema

**Step 2: Define Data Types and Domain of Attributes.**

Product(<u>PName</u>, Price, Category, Manfacturer)

<u>Pname : Varchar</u>,

Price: Float,

Category: Varchar

Manfacturer: Varchar

## Step 3: Specifying Constraints.

Product(<u>PName</u>, Price, Category, Manfacturer)

**Constraints: Restrictions on values of Attribute.**

- Specifying Attribute and Domain Constraints
- Specifying Key Constraints
- Specifying Key and Referential Integrity Constraints

## Specifying Attribute and Domain Constraints

- **NOT NULL**
  - ✓NULL is not permitted for a particular attribute
- **Default value**
  - ✓DEFAULT <value>
- **CHECK** clause
  - ✓Dnumber > 0 AND Dnumber < 21;
- **UNIQUE** clause
  - ✓Specifies attributes that have unique values

## Specifying Key Constraints

- **PRIMARY KEY** clause
  - ✓ Specifies one or more attributes that make up the primary key of a relation
  - ✓ **It is an attribute or a combination of attributes that  that uniquely identifies the records./tuples**

e.g. roll_no, account_no, Id etc.

**PRIMARY KEY = NOT NULL+ UNIQUE**

## Schema of Table Product

**Product**(Pname varchar Primary Key,
Price float Not Null,
Category varchar, check(Gadget, Photoraphy, Household
Manufacturer varchar )

| Attribute | Data Type | Constraints |
|---|---|---|
| Pname | Varchar | Primary Key |
| Price | Float | Not Null |
| Category | Varchar | Gadget, Photography, Household |
| Manufacturer | Varchar | |

## LET’S CODE TOGETHER!!

# Creating a Database

## Step 1. Create a Database Company

```
CREATE DATABASE  <DATABSE NAME>;
```

Create database company;

## Step 2. USE Database

```
USE  <DATABSE NAME>;
```

use company;

## Step 2. SHOW TABLES

show tables;

# Creating a Table

## Step 1. Create a TABLE

```
CREATE TABLE  <TABLE NAME> (
<ATTRIBUTE LIST> <DATA TYPE>
<CONSTRAINT>,
<ATTR2> <DATA TYPE> <CONSTRAINT>);
```

| Attribute | Data Type | Constraints |
|---|---|---|
| Pname | Varchar | Primary Key |
| Price | Float | Not Null |
| Category | Varchar | Gadget, Photography, Household |
| Manufacturer | Varchar | |

## Creating a Table

create table product(Pname varchar(20) primary key,
price float NOT NULL,category varchar(20)
CHECK(category
in("Gadget","Photography","Household")),
manufacturer varchar(20));

| Attribute | Data Type | Constraints |
|---|---|---|
| Pname | Varchar | Primary Key |
| Price | Float | Not Null |
| Category | Varchar | Gadget, Photography, Household |
| Manufacturer | Varchar | |

## Show Existing Tables

```
Show tables;
```

## Describe structure of a Existing Table

```
Desc <tablename>;
```

Desc product;

# Insert records in Table

```
INSERT  INTO  R(A1,…., An)  VALUES  (v1,…., vn)
```

```
insert into product(Pname,price,category,manufacturer)
    values("Gizmo",19.99, "Gadgets", "GizmoWorks");
```

or

```
insert into product values("Gizmo",19.99, "Gadgets", "GizmoWorks");
insert into product values("Powergizmo",29.99, "Gadgets", "GizmoWorks");
insert into product values("SingleTouch",149.99, "Photography", "Canon");
insert into product values("MultiTouch",203.99, "Household", "Hitachi");
```

| PName | Price | Category | Manufacturer |
|---|---|---|---|
| Gizmo | 19.99 | Gadgets | GizmoWorks |
| Powergizmo | 29.99 | Gadgets | GizmoWorks |
| SingleTouch | 149.99 | Photography | Canon |
| MultiTouch | 203.99 | Household | Hitachi |

## Select Query

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

```
SELECT  *
FROM    product;
```

“selection”

| PName | Price | Category | Manufacturer |
|---|---|---|---|
| Gizmo | 19.99 | Gadgets | GizmoWorks |
| Powergizmo | 29.99 | Gadgets | GizmoWorks |
| SingleTouch | 149.99 | Photography | Canon |
| MultiTouch | 203.99 | Household | Hitachi |

## Select Query using WHERE

Product

| PName | Price | Category | Manufacturer |
|---|---|---|---|
| Gizmo | 19.99 | Gadgets | GizmoWorks |
| Powergizmo | 29.99 | Gadgets | GizmoWorks |
| SingleTouch | 149.99 | Photography | Canon |
| MultiTouch | 203.99 | Household | Hitachi |

```
SELECT  Pname, Price
FROM    Product
```

“projection”

| PName | Price |
|---|---|
| Gizmo | 19.99 |
| Powergizmo | 29.99 |
| SingleTouch | 149.99 |
| MultiTouch | 203.99 |

## Select Query using WHERE

Product

| PName | Price | Category | Manufacturer |
|---|---|---|---|
| Gizmo | 19.99 | Gadgets | GizmoWorks |
| Powergizmo | 29.99 | Gadgets | GizmoWorks |
| SingleTouch | 149.99 | Photography | Canon |
| MultiTouch | 203.99 | Household | Hitachi |

```
SELECT   *
FROM     Product
WHERE    category=‘Gadgets’;
```

“selection” with where

| PName | Price | Category | Manufacturer |
|---|---|---|---|
| Gizmo | 19.99 | Gadgets | GizmoWorks |
| Powergizmo | 29.99 | Gadgets | GizmoWorks |

# Select Query using WHERE

Product

| PName | Price | Category | Manufacturer |
|---|---|---|---|
| Gizmo | 19.99 | Gadgets | GizmoWorks |
| Powergizmo | 29.99 | Gadgets | GizmoWorks |
| SingleTouch | 149.99 | Photography | Canon |
| MultiTouch | 203.99 | Household | Hitachi |

```
SELECT   PName, Price, Manufacturer
FROM     Product
WHERE    Price > 100;
```

$\Downarrow$

| PName | Price | Manufacturer |
|---|---|---|
| SingleTouch | 149.99 | Canon |
| MultiTouch | 203.99 | Hitachi |

“selection” and “projection” with where

# Select Query using WHERE

Product

| PName | Price | Category | Manufacturer |
|---|---|---|---|
| Gizmo | 19.99 | Gadgets | GizmoWorks |
| Powergizmo | 29.99 | Gadgets | GizmoWorks |
| SingleTouch | 149.99 | Photography | Canon |
| MultiTouch | 203.99 | Household | Hitachi |

```
SELECT   PName, Price, Manufacturer
FROM     Product
WHERE    Price > 100 and  manufacturer =“Canon”;
```

Combine two or more conditions Using and

| PName | Price | Manufacturer |
|---|---|---|
| SingleTouch | 149.99 | Canon |

# Select Query using WHERE

Product

| PName | Price | Category | Manufacturer |
|---|---|---|---|
| Gizmo | 19.99 | Gadgets | GizmoWorks |
| Powergizmo | 29.99 | Gadgets | GizmoWorks |
| SingleTouch | 149.99 | Photography | Canon |
| MultiTouch | 203.99 | Household | Hitachi |

```
SELECT   PName, Price, Manufacturer
FROM     Product
WHERE    manufacturer =“Hitachi” or
manufacturer = “Canon”;
```

Combine two or more conditions Using or

| PName | Price | Manufacturer |
|---|---|---|
| SingleTouch | 149.99 | Canon |
| MultiTouch | 203.99 | Hitachi |

# Select Query using WHERE

Product

| PName | Price | Category | Manufacturer |
|---|---|---|---|
| Gizmo | 19.99 | Gadgets | GizmoWorks |
| Powergizmo | 29.99 | Gadgets | GizmoWorks |
| SingleTouch | 149.99 | Photography | Canon |
| MultiTouch | 203.99 | Household | Hitachi |

```
SELECT   PName, Price, Manufacturer
FROM     Product
WHERE    manufacturer  IN(“Hitachi”,“Canon”);
```

| PName | Price | Manufacturer |
|---|---|---|
| SingleTouch | 149.99 | Canon |
| MultiTouch | 203.99 | Hitachi |

Replace OR with In conditions Using IN

## Note That

- Case insensitive:
  - Same: SELECT  Select  select
  - Same: Product   product
  - Different: ‘Seattle’  ‘seattle’

- Constants:
  - ‘abc’  - yes
  - “abc” - no

## The LIKE operator

```
SELECT   *
FROM     Products
WHERE    PName LIKE  <pattern>
```

**Pattern :** pattern matching on strings. It contains two special symbols:

- %  = any sequence of characters
- _   = any single character

## Like Operator with %

Product name that starts with P

Product

| PName | Price | Category | Manufacturer |
|---|---|---|---|
| Gizmo | 19.99 | Gadgets | GizmoWorks |
| Powergizmo | 29.99 | Gadgets | GizmoWorks |
| SingleTouch | 149.99 | Photography | Canon |
| MultiTouch | 203.99 | Household | Hitachi |

```
SELECT   *
FROM     Product
WHERE    Pname like ‘p%’;
```

| PName | Price | Category | Manufacturer |
|---|---|---|---|
| Powergizmo | 29.99 | Gadgets | GizmoWorks |

## Like Operator with %

Product name that ends with Touch

Product

| PName | Price | Category | Manufacturer |
|---|---|---|---|
| Gizmo | 19.99 | Gadgets | GizmoWorks |
| Powergizmo | 29.99 | Gadgets | GizmoWorks |
| SingleTouch | 149.99 | Photography | Canon |
| MultiTouch | 203.99 | Household | Hitachi |

```
SELECT   *
FROM     Product
WHERE    Pname like ‘%Touch’;
```

| PName | Price | Category | Manufacturer |
|---|---|---|---|
| SingleTouch | 149.99 | Photography | Canon |
| MultiTouch | 203.99 | Household | Hitachi |

## Like Operator with %

Product name that contains e anywhere in the name

Product

| PName | Price | Category | Manufacturer |
|---|---|---|---|
| Gizmo | 19.99 | Gadgets | GizmoWorks |
| Powergizmo | 29.99 | Gadgets | GizmoWorks |
| SingleTouch | 149.99 | Photography | Canon |
| MultiTouch | 203.99 | Household | Hitachi |

```
SELECT   *
FROM     Product
WHERE    Pname like ‘%e%’;
```

| PName | Price | Category | Manufacturer |
|---|---|---|---|
| Powergizmo | 29.99 | Gadgets | GizmoWorks |
| SingleTouch | 149.99 | Photography | Canon |

## Like Operator with _ &%

Product name with second letter ‘o’

Product

| PName | Price | Category | Manufacturer |
|---|---|---|---|
| Gizmo | 19.99 | Gadgets | GizmoWorks |
| Powergizmo | 29.99 | Gadgets | GizmoWorks |
| SingleTouch | 149.99 | Photography | Canon |
| MultiTouch | 203.99 | Household | Hitachi |

```
SELECT   *
FROM     Product
WHERE    Pname like ‘_o%’;
```

| PName | Price | Category | Manufacturer |
|---|---|---|---|
| Powergizmo | 29.99 | Gadgets | GizmoWorks |

## Like Operator with %

Product name with second last character ‘c’

Product

| PName | Price | Category | Manufacturer |
|---|---|---|---|
| Gizmo | 19.99 | Gadgets | GizmoWorks |
| Powergizmo | 29.99 | Gadgets | GizmoWorks |
| SingleTouch | 149.99 | Photography | Canon |
| MultiTouch | 203.99 | Household | Hitachi |

```
SELECT   *
FROM     Product
WHERE    Pname like ‘%c_’;
```

| PName | Price | Category | Manufacturer |
|---|---|---|---|
| SingleTouch | 149.99 | Photography | Canon |
| MultiTouch | 203.99 | Household | Hitachi |

## Eliminating Duplicates

```
SELECT   DISTINCT category
FROM     Product;
```

| Category |
|---|
| Gadgets |
| Photography |
| Household |

Compare to:

```
SELECT   category
FROM     Product;
```

| Category |
|---|
| Gadgets |
| Gadgets |
| Photography |
| Household |

## Aggregate Functions

SQL supports several aggregation operations:

- ✓ Sum
- ✓ Max
- ✓ Min
- ✓ Avg
- ✓ Count

Except count, all aggregations apply to a single attribute

## Aggregate Functions – SUM

Sum of Price of all Products

Product

| PName | Price | Category | Manufacturer |
|---|---|---|---|
| Gizmo | 19.99 | Gadgets | GizmoWorks |
| Powergizmo | 29.99 | Gadgets | GizmoWorks |
| SingleTouch | 149.99 | Photography | Canon |
| MultiTouch | 203.99 | Household | Hitachi |

```
SELECT   sum(price)
FROM      Product;
```

```
403.96
```

# Aggregate Functions – MAX

Max of Price of all Products

Product

| PName | Price | Category | Manufacturer |
|---|---|---|---|
| Gizmo | 19.99 | Gadgets | GizmoWorks |
| Powergizmo | 29.99 | Gadgets | GizmoWorks |
| SingleTouch | 149.99 | Photography | Canon |
| MultiTouch | 203.99 | Household | Hitachi |

```
SELECT   max(price)
FROM     Product;
```

```
203.96
```

# Aggregate Functions – MIN

Min of Price of all Products

Product

| PName | Price | Category | Manufacturer |
|---|---|---|---|
| Gizmo | 19.99 | Gadgets | GizmoWorks |
| Powergizmo | 29.99 | Gadgets | GizmoWorks |
| SingleTouch | 149.99 | Photography | Canon |
| MultiTouch | 203.99 | Household | Hitachi |

```
SELECT   min(price)
FROM     Product;
```

```
19.99
```

# Aggregate Functions – AVG

## Avg of Price of all Products

Product

| PName | Price | Category | Manufacturer |
|---|---|---|---|
| Gizmo | 19.99 | Gadgets | GizmoWorks |
| Powergizmo | 29.99 | Gadgets | GizmoWorks |
| SingleTouch | 149.99 | Photography | Canon |
| MultiTouch | 203.99 | Household | Hitachi |

```
SELECT   avg(price)
FROM     Product;
```

$\Downarrow$

```
100.99
```

# Aggregate Functions – COUNT

Total number of Products

Product

| PName | Price | Category | Manufacturer |
|---|---|---|---|
| Gizmo | 19.99 | Gadgets | GizmoWorks |
| Powergizmo | 29.99 | Gadgets | GizmoWorks |
| SingleTouch | 149.99 | Photography | Canon |
| MultiTouch | 203.99 | Household | Hitachi |

```
SELECT   count(price)
FROM      Product;
```

```
SELECT   count(*)
FROM      Product;
```

```
4
```

## More Examples

| Query | Sql |
|---|---|
| Max price of Gadgets category Products | Select Max(price) from product where category=“Gadgets” |
| Total no of products in Household category | Select count(*) from product where Category=“Household” |
| Count total no. of categories | Select Count(Distinct(category) ) from product |

## WRITE THE QUERY

| PName | Price | Category | Manufacturer |
|---|---|---|---|
| Gizmo | 19.99 | Gadgets | GizmoWorks |
| Powergizmo | 29.99 | Gadgets | GizmoWorks |
| SingleTouch | 149.99 | Photography | Canon |
| MultiTouch | 203.99 | Household | Hitachi |

| Problem Statement | SQL Query |
|---|---|
| Average Price of Gizmo Works manufacturer | ? |
| Total price of Gizmo Works manufacturer | ? |
| Count total number of manufacturers | ? |
| Count number of products that contains ‘o’ in their name | ? |

## Ordering the Results

```
SELECT   pname, price, manufacturer
FROM     Product
WHERE    manufacturer='GizmoWorks' AND price > 50
ORDER BY  price, pname;
```

- Ties are broken by the second attribute on the ORDER BY list, etc.

- Also works without Where
- Ordering is ascending, unless you specify the DESC keyword.

```
SELECT   pname, price, manufacturer
FROM     Product
ORDER BY  price DESC;
```

## FIND THE RESULT

| PName | Price | Category | Manufacturer |
|---|---|---|---|
| Gizmo | 19.99 | Gadgets | GizmoWorks |
| Powergizmo | 29.99 | Gadgets | GizmoWorks |
| SingleTouch | 149.99 | Photography | Canon |
| MultiTouch | 203.99 | Household | Hitachi |

```
SELECT   DISTINCT category
FROM     Product
ORDER BY category
```

⇨ ?

```
SELECT   Category
FROM     Product
ORDER BY  PName
```

⇨ ?

```
SELECT   DISTINCT category
FROM     Product
ORDER BY PName
```

⇨ ?

## Practice Exercise

## Create a new table in your current database ‘COMPANY’ with the following schema

| Attribute | Data Type | Constraints |
|---|---|---|
| Cname | Varchar | Primary Key |
| Reg_Date | Date | Not Null |
| Stock_Price | Float | |
| Country | Varchar | |

## Create a new table named ‘COMPDTLS’ in your current database with the following schema

| Attribute | Data Type | Constraints |
|---|---|---|
| CompName | Varchar | Primary Key |
| RegDate | Date | Not Null |
| StockPrice | Float | |
| Country | Varchar | |

```
COMPDTLS(CompName   varchar Primary Key,
         RegDate    Date Not Null,
         StockPrice Float
         Country    varchar   )
```

## Insert the following Records in COMPDTLS

| CompName | RegDate | StockPrice | Country |
|---|---|---|---|
| GizmoWorks | 2019/10/21 | 25 | USA |
| Canon | 2019/10/3 | 65 | Japan |
| Hitachi | 2019/10/10 | 15 | India |

## Write SQL Queries for:

1. List the details of all companies
2. List the registration date of all companies
3. Show the details of all companies of Japan
4. List  the company name whose stock price is 65
5. List the companies of Japan or India
6. Show the maximum stock price.
7. Show the average stock price.
8. Show the distinct countries
9. Show the total no of countries
10. Show the company name whose country name ends with  ‘a’.