Python Programming

Day 15: *Introduction to Database*

Introduction to Database

Insert

Delete

Select

Update

Color and symbol meaning

❗	Hint
✓	Preferred
❓	Student's activity
⬜	Practice code

🟨	Keyword
🟪	In-built functions
🟩	Strings
🟦	Output

Database Definition

A database is simply an organized collection of related data, typically stored on disk, and accessible by possibly many concurrent users.

Databases are generally separated into application areas. For example, one database may contain Human Resource (employee and payroll) data; another may contain sales data; another may contain accounting data; and so on. Databases are managed by a DBMS.

Database Definition

Types of Database

- ❖ **Relational database (MySQL, Oracle, MSSQL etc.)**
- ❖ Flat-file database (Excel, Notepad, csv etc.)
- ❖ NoSQL (MongoDB, couch DB etc.)
- ❖ Object-oriented database (Objectivity DB, VelocityDB etc)
- ❖ Object-relational database (PostgreSQL, Oracle etc)

Database Management System (DBMS)

A Database Management System (DBMS) is a piece of software designed to store and manage databases

Example

- **MySQL**
- **Oracle**
- **PostgreSQL**
- **Microsoft Access**
- **SQL Server**

Data model

A data model is a collection of concepts for describing data

The relational data model is the most widely used model today.

Its main Concept is the relation; which is essentially, a **table**

Data model

A **schema** is a description of a particular collection of data, using the given data model

The **schema** of a table is the table name, its attributes, and their types

Example

Product(Pname: *string*, Price: *float*, Category: *string*, Manufacturer: *string*)

Data model

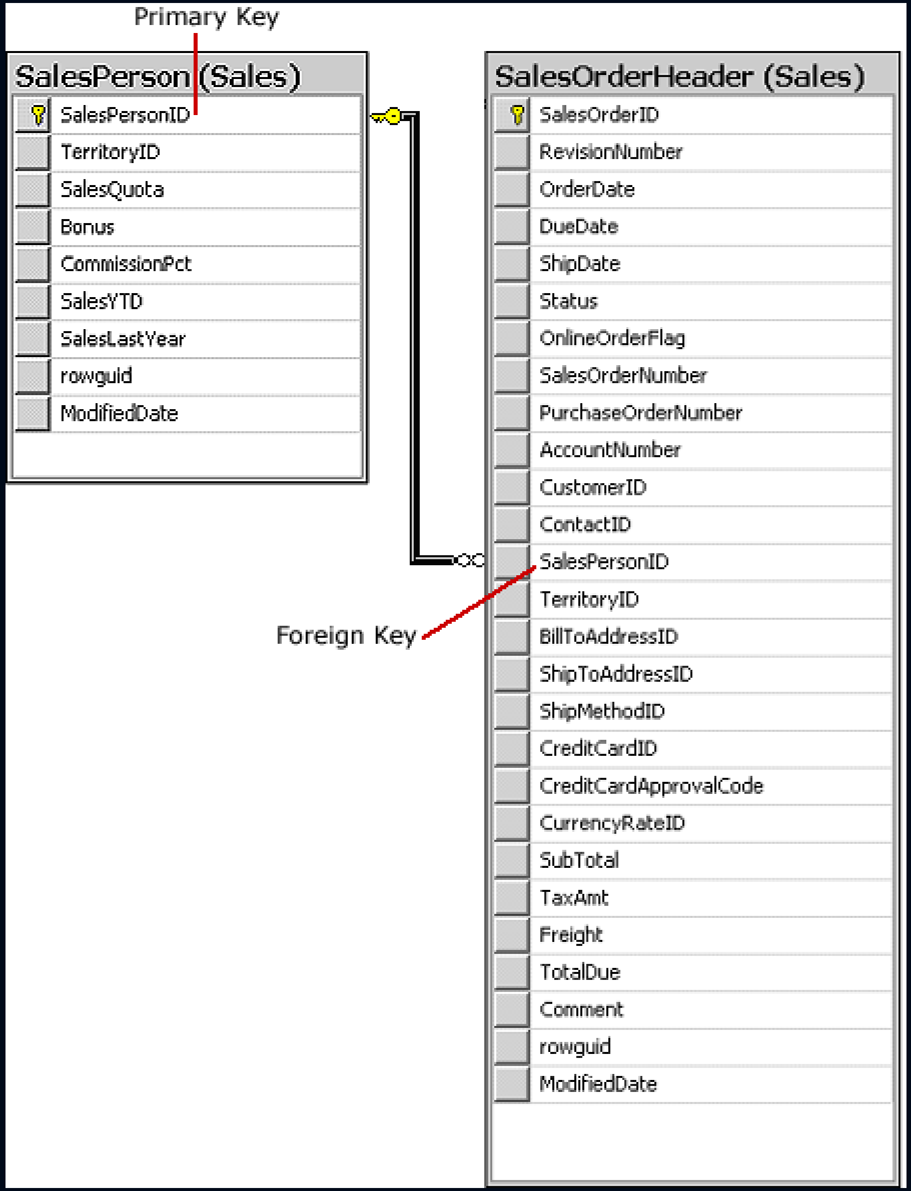
A Primary **key** is a minimal subset of attributes that acts as a unique identifier for tuples in a relation

A primary **key** is an attribute whose values are unique; we underline a key.

Product(Pname: *string*, Price: *float*, Category: *string*, Manufacturer: *string*)

Data model

A **Foreign key** is a field (or collection of fields) in one table that uniquely identifies a row of another table



SQL Introdution

SQL is a standard language for querying and manipulating data

SQL stands for

- ***Structured***
- ***Query***
- ***Language***

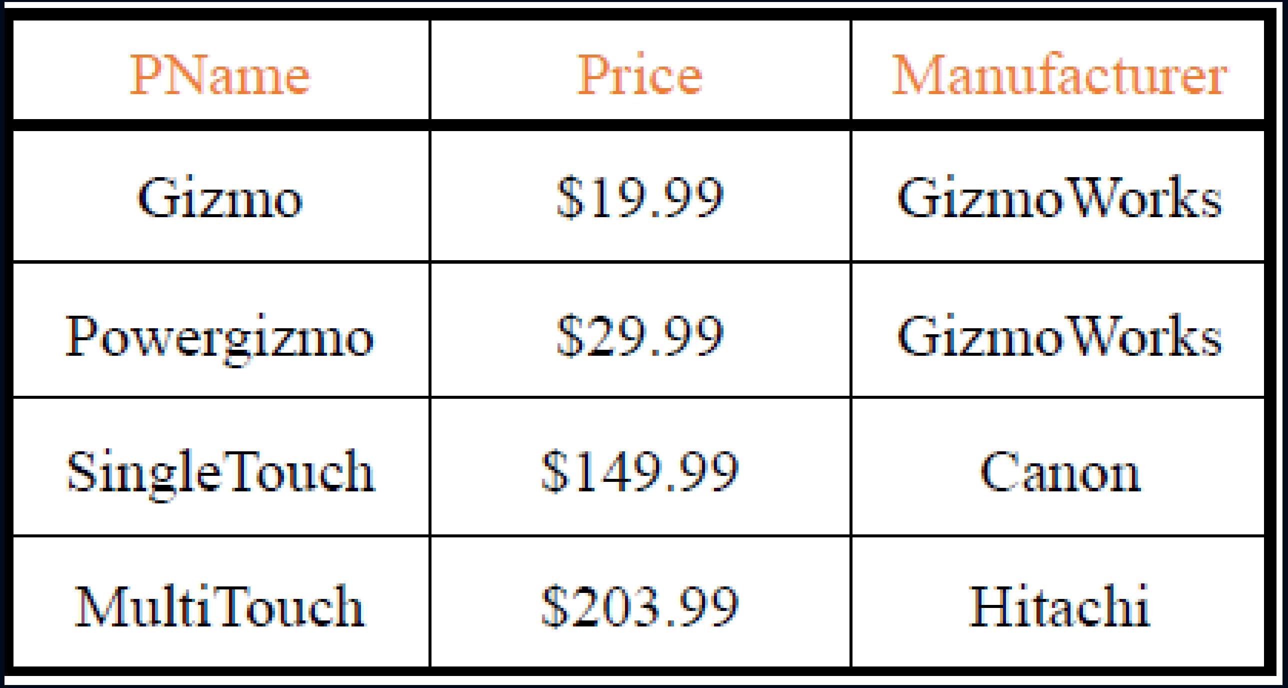
SQL Introdution

SQL is a…

- ❑ Data Definition Language (DDL)
  - Define relational schemata
  - Create/alter/delete tables and their attributes
- ❑ Data Manipulation Language (DML)
  - Insert/delete/modify tuples in tables
  - Query one or more tables – discussed next!

Tables in SQL

Product

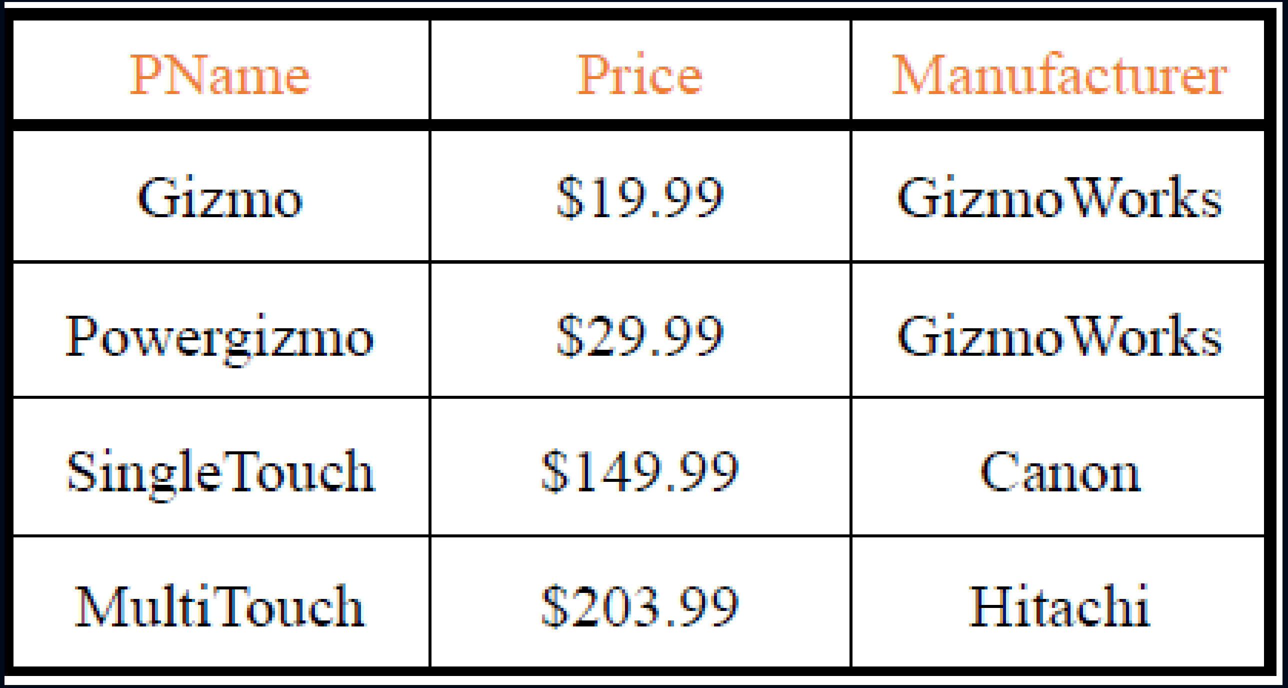


PName	Price	Manufacturer
Gizmo	$19.99	GizmoWorks
Powergizmo	$29.99	GizmoWorks
SingleTouch	$149.99	Canon
MultiTouch	$203.99	Hitachi

A relation or table is a multiset of tuples having the attributes specified by the schema

Tables in SQL

Product



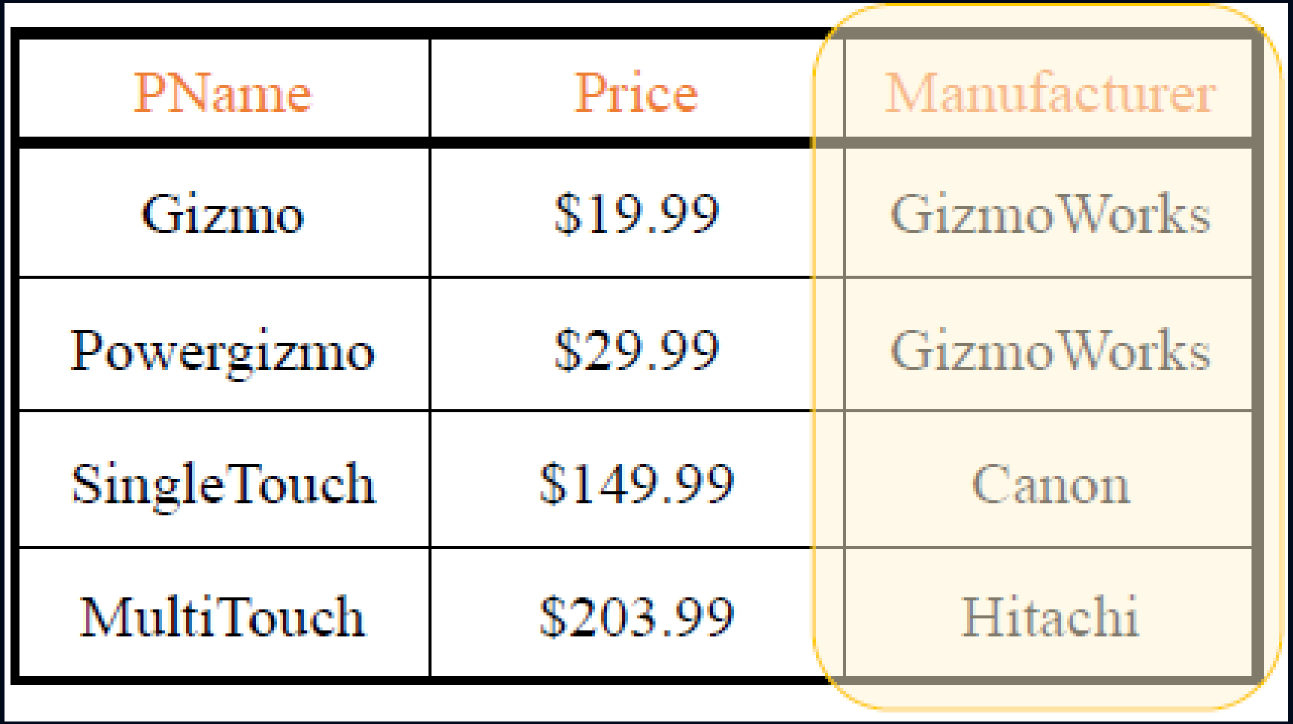
PName	Price	Manufacturer
Gizmo	$19.99	GizmoWorks
Powergizmo	$29.99	GizmoWorks
SingleTouch	$149.99	Canon
MultiTouch	$203.99	Hitachi

A multiset is an unordered list (or: a set with multiple duplicate instances allowed)

List: [1, 1, 2, 3]
Set: {1, 2, 3}
Multiset: {1, 1, 2, 3}

Tables in SQL

Product



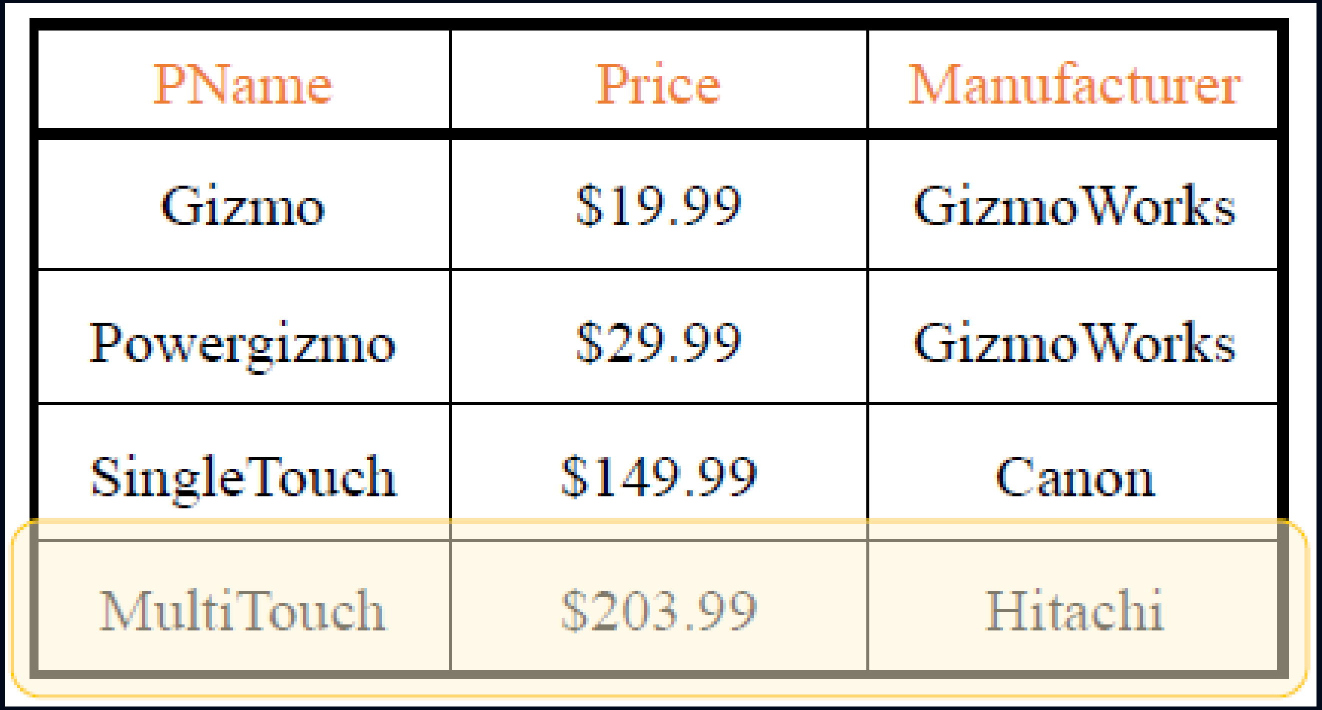
PName	Price	Manufacturer
Gizmo	$19.99	GizmoWorks
Powergizmo	$29.99	GizmoWorks
SingleTouch	$149.99	Canon
MultiTouch	$203.99	Hitachi

An attribute (or column) is a typed data entry present in each tuple in the relation

NB: Attributes must have an **atomic type** in standard SQL, i.e. **not a list, set,** etc.

Tables in SQL

Product



PName	Price	Manufacturer
Gizmo	$19.99	GizmoWorks
Powergizmo	$29.99	GizmoWorks
SingleTouch	$149.99	Canon
MultiTouch	$203.99	Hitachi

A tuple or row is a single entry in the table having the attributes specified by the schema

Also referred to as a record

Data Types in SQL

Atomic types:

- **Characters: CHAR(20), VARCHAR(50)**
- **Numbers: INT, BIGINT, SMALLINT, FLOAT**
- **Others: MONEY, DATETIME, …**

- **Every attribute must have an atomic type**
  - **Hence tables are flat**

SQLite

- However, we shall use **SQLite** database for the course of this training.
- **SQLite** comes with **python 3** therefore no need for installation.
- We shall use **"DB browser for SQLite"** to access the database.
- Download from http://sqlitebrowser.org/

SQLite

SQLite has 5 basic data types these includes;

- **Text - string**
- **Integer**
- **Real – floating point numbers**
- **Blob – Binary Data (images, audio, multimedia)**
- **Numeric**

Connecting to SQLite from Python

```
import sqlite3

conn = sqlite3.connect('test.db')
c = conn.cursor()
```

The **import statement** makes **sqlite** methods **accessible**

The **connect method** establishes a **connection** to the **database** if available else it **creates a new database**.

The **cursor** is an object created by connection to **handle all executions** related to the connection.

Create a DB Table

```
def create_table():
   c.execute('CREATE TABLE IF NOT
EXISTS test(id INTEGER PRIMARY
KEY, name TEXT)')


create_table()
```

After a connection has been established we can create tables and perform basic CRUD operations (CREATE, READ, UPDATE, DELETE)

Insert Record

Single-table queries

```
INSERT INTO <tableName> (column1, column2, column3, ...)
VALUES (value1, value2, value3, ...)
```

```
INSERT INTO <tableName>
VALUES (value1, value2, value3, ...)
```

If you are adding values for all the columns of the table, you do not need to specify the column names in the SQL query. However, make sure the order of the values is in the same order as the columns in the table.

Insert Record

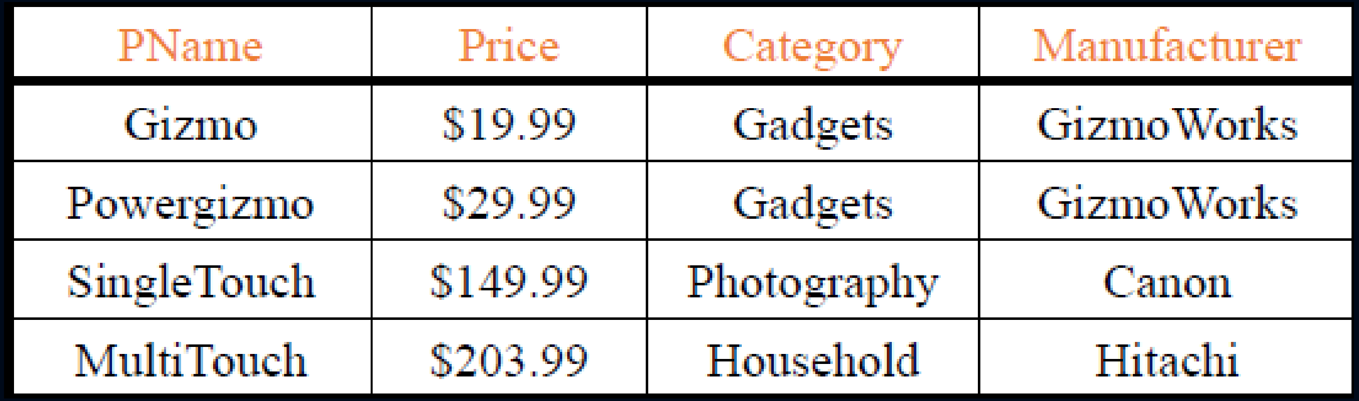
```
def create_table():
   c.execute('CREATE TABLE IF NOT EXISTS
product(PName TEXT, Price TEXT, category TEXT,
manufacturer TEXT)')
def data_entry():
   c.execute("INSERT INTO category VALUES (?,?,?,?)”,
(‘Gizmo’, 3000, ‘Gadgets’, ‘Gizmo Works’))
   conn.commit()
   c.close()
   conn.close()

create_table()
data_entry()
```

The data_entry function executes the **INSERT** statement. In **MySQL** ‘%s’ is used instead of ‘?’

The **commit** method makes the changes made to the table **IRREVERSIBLE**

Class Activity 1

Create the following table and insert the records into the table using python.



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

Reading records from the DB

SELECT <attributes>

FROM <one or more relations>

WHERE <conditions>

Call this a SFW query.

Reading records from the DB

Simple SQL Query: Selection

Selection is the operation of **filtering** a relation's tuples on some condition

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'
```

PName	Price	Category	Manufacturer
Gizmo	$19.99	Gadgets	GizmoWorks
Powergizmo	$29.99	Gadgets	GizmoWorks

Reading records from the DB

A Few Details

- SQL commands are case insensitive:
  - Same: SELECT, Select, select
  - Same: Product, product
- Values are not (i.e. case sensitive):
  - Different: ‘Seattle’ ≠ ‘seattle’
- Use single quotes for constants:
  - ‘abc’ - yes
  - “abc” - no

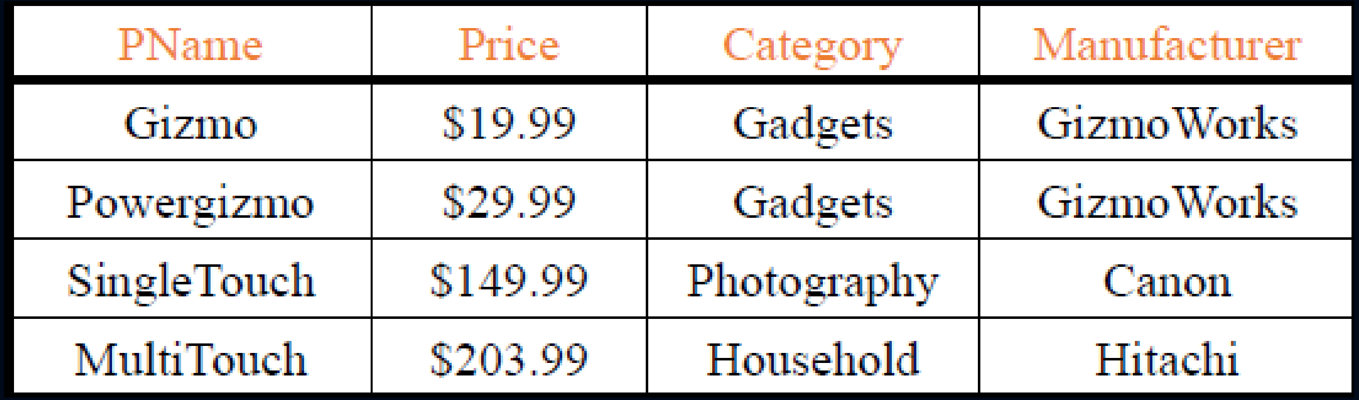
Reading Records from the DB

```
def read_data():
  c.execute("SELECT * FROM product")
  for row in c.fetchall():
    print (row)
```

The **fetchall()** is used to **get all** data returned by the query. There exist **fetchone()** that returns only **one** record from the query result

Class Activity 2

Find all products under $200 manufactured in Japan;
return their names and prices



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

Class Activity 3

From product table Select Pname and Price for **Category = ‘Gadgets’**

Update Record

```
UPDATE <tableName>
SET column1 = value1, column2 = value2, ...
WHERE <conditions>
```

Note: Be careful when updating records in a table! Notice the WHERE clause in the UPDATE statement. The WHERE clause specifies which record(s) that should be updated. If you omit the WHERE clause, all records in the table will be updated!

Update Record

```
def update_data():
    c.execute("UPDATE product SET
price = ? WHERE Pname = ?",
('$50.33', 'Gizmo'))
    conn.commit()
```

The **UPDATE** statement is used to **modify** a record in a database

Delete Record

DELETE FROM <tableName>

WHERE <conditions>

Note: Be careful when deleting records in a table! Notice the WHERE clause in the DELETE statement. The WHERE clause specifies which record(s) that should be deleted. If you omit the WHERE clause, all records in the table will be deleted!

Delete Record

```
def delete_data():
    c.execute("DELETE FROM product
WHERE PName = ?", ('Gizmo'))
    conn.commit()
```

The **DELETE** statement is used to **modify** a record in a database.

Class Activity 4

Update the “Gizmo works” value in the manufacturer column to “Sony” and Delete record with “Powergizmo” as PName.

Next Lecture ...

Python Programming

Day 16: *Introduction to Database (2)*