Python Summer Training

File I/O

Python Team, BFCAI

File I/O

- A file is a container in computer storage devices used for storing data.
- When we want to read from or write to a file, we need to open it first.
- When we are done, it needs to be closed so that the resources that are tied with the file are freed.
- Hence, in Python, a file operation takes place in the following order:
  1. Open a file
  2. Read or write (perform operation)
  3. Close the file

File I/O: Openning Files

- In Python, we use the `open()` method to open files.
- Here's few simple examples of how to open a file in different modes.

```
file1 = open("data.txt", "r")        # read in text mode
file2 = open("data.txt",'w')         # write in text mode
file3 = open("data.txt",'a')         # append in text mode
```

File I/O: Reading Files

```
# Open a file named `data.txt`
file = open('data.txt', 'r')

# Read all file contents
contents = file.read()

# Print file contents
print(contents)

# Close the file
file.close()
```

File I/O: Reading Files

```
# Open a file named `data.txt`
file = open('data.txt', 'r')

# Read all lines in the file
lines = file.readlines()

# Print the lines
for line in lines:
    print(line.strip())

# Close the file
file.close()
```

File I/O: Writing to Files

```
# Open a file named `data.txt`
file = open('data.txt', 'w')

# Write contents to the file
file.write('My name is Ahmed.')
file.write('I Love BFCAI.')

# Close the file
file.close()
```

File I/O: Writing to Files

```
# Open a file named `data.txt`
file = open('data.txt', 'w')

# Write contents to the file
file.write('My name is Ahmed.\n')
file.write('I Love BFCAI.\n')

# Close the file
file.close()
```

File I/O: Appending to Files

```
# Open a file named `data.txt`
file = open('data.txt', 'a')

# Append contents to the file
file.write('My name is Ahmed.\n')
file.write('I Love BFCAI.\n')

# Close the file
file.close()
```

Python Summer Training

Exception Handling

Python Team, BFCAI

Exceptions: Simple Code

```
num1 = int(input('Enter a number: '))
num2 = int(input('Enter a number: '))

result = num1 / num2
print(result)
print('Thank you for using our calculator.')

# Run
Enter a number: 10
Enter a number: 2
5.0
Thank you for using our calculator.
```

Exceptions: Simple Code

```
num1 = int(input('Enter a number: '))
num2 = int(input('Enter a number: '))

result = num1 / num2
print(result)
print('Thank you for using our calculator.')

# Run
Enter a number: 10
Enter a number: 0

Error in line result = num1 / num2
ZeroDivisionError: division by zero
```

Exceptions: Simple Code

```
num1 = int(input('Enter a number: '))
num2 = int(input('Enter a number: '))

result = num1 / num2
print(result)
print('Thank you for using our calculator.')

# Run
Enter a number: 10
Enter a number: ahmed

Error in line num2 = int(input('Enter a number: '))
ValueError: invalid literal for int() with base 10: 'ahmed'
```

Exceptions

- An exception is an unexpected event that occurs during program execution.
- For example, the following code causes an exception as it is not possible to divide a number by 0.

```
divide_by_zero = 7 / 0
```

- Errors that occur at runtime (after passing the syntax test) are called exceptions.
- Whenever these runtime errors occur, Python creates an exception object.
- If not handled properly, it prints a traceback to that error along with some details about why that error occurred.

Exceptions

For instance, they occur when we

- Try to divide a number by zero (`ZeroDivisionError`)
- Try to open a file that does not exist (`FileNotFoundError`)
- Try to import a module that does not exist (`ImportError`).

Exception Handling

- The `try...except` block is used to handle exceptions in Python.

```
try:
    # code that may cause exception
except:
    # code to run when exception occurs
```

- We place the code that might generate an exception inside the `try` block.
- When an exception occurs, it is caught by the `except` block.
- The `except` block cannot be used without the `try` block.

Exception Handling: Simple Code

```
try:
    num1 = int(input('Enter a number: '))
    num2 = int(input('Enter a number: '))

    result = num1 / num2
    print(result)
    print('Thank you for using our calculator.')
except:
    print('Please enter a valid inputs.')

# Run
Enter a number: 10
Enter a number: 2
5.0
Thank you for using our calculator.
```

Exception Handling: Simple Code

```
try:
    num1 = int(input('Enter a number: '))
    num2 = int(input('Enter a number: '))

    result = num1 / num2
    print(result)
    print('Thank you for using our calculator.')
except:
    print('Please enter a valid inputs.')

# Run
Enter a number: 10
Enter a number: 0
Please enter a valid inputs.
```

Exception Handling: Simple Code

```
try:
    num1 = int(input('Enter a number: '))
    num2 = int(input('Enter a number: '))

    result = num1 / num2
    print(result)
    print('Thank you for using our calculator.')
except:
    print('Please enter a valid inputs.')

# Run
Enter a number: 10
Enter a number: ahmed
Please enter a valid inputs.
```

Exception Handling: try … finally

- The `finally` block is always executed no matter whether there is an exception or not.

```
try:
    # code that may cause exception
except:
    # code to run when exception occurs
finnaly:
    # code that always run
```

Exception Handling: try … finally

```
try:
    num1 = int(input('Enter a number: '))
    num2 = int(input('Enter a number: '))

    result = num1 / num2
    print(result)
except:
    print('Please enter a valid inputs.')
finally:
    print('Thank you for using our calculator.')

# Run
Enter a number: 10
Enter a number: 2
5.0
Thank you for using our calculator.
```

Exception Handling: try … finally

```
try:
    num1 = int(input('Enter a number: '))
    num2 = int(input('Enter a number: '))

    result = num1 / num2
    print(result)
except:
    print('Please enter a valid inputs.')
finally:
    print('Thank you for using our calculator.')

# Run
Enter a number: 10
Enter a number: 0
Please enter a valid inputs.
Thank you for using our calculator.
```

Python Summer Training

Database Programming

Python Team, BFCAI

Database Jobs in Egypt

Senior Oracle Database Developer

Ejada - Cairo, Egypt

4 days ago

Full Time

Experienced · 5 - 10 Yrs of Exp · Banking · IT/Software Development· Information Technology (IT) · **Database**· Oracle

Database Administrator

Perfect Presentation - 6th of October, Giza, Egypt

3 days ago

Full Time

Experienced · 8 - 12 Yrs of Exp · IT/Software Development · **Database** Administrator · **Database**· Oracle· SQL · Information Technology (IT)· Computer Science· Administrator

Microsoft SQL Database Senior Administrator

Premier Services and Recruitment - 6th of October, Giza, Egypt

8 days ago

Full Time

Manager · 6 - 10 Yrs of Exp · IT/Software Development· SQL· Administrator· Microsoft SQL · SQL **Database**

Database Jobs in Egypt

Oracle Database Administrator

HR International Business - 6th of October, Giza, Egypt

29 days ago

Full Time

Entry Level · 1 - 5 Yrs of Exp · IT/Software Development· ORACLE ADMINISTRATION· Oracle DBA· Oracle· DBA · Information Technology (IT)· SQL· OCP· OCA

GIS database Administrator

Perfect Presentation - 6th of October, Giza, Egypt

1 month ago

Full Time

Experienced · 5 - 7 Yrs of Exp · IT/Software Development· Software Development· GIS· Software Engineering· SQL

Oracle Database Administrator

Easy System - New Capital, Cairo, Egypt

2 months ago

Full Time **Part Time**

Experienced · 2+ Yrs of Exp · Installation/Maintenance/Repair · IT/Software Development · Engineering - Telecom/Technology · Oracle · **Database**· Linux· developer· Information Technology (IT)· Computer Science· Software Development

Database Jobs in Egypt

Microsoft SQL Database Senior Administrator

Premier Services and Recruitment - 6th of October, Giza, Egypt

8 days ago

Full Time

Manager · 6 - 10 Yrs of Exp · IT/Software Development · **SQL**· Administrator · Microsoft **SQL** · **SQL** Database

SQL Database Lead

Ibn Sina Pharma - New Cairo, Cairo, Egypt

16 days ago

Full Time

Experienced · 5 - 7 Yrs of Exp · Installation/Maintenance/Repair · IT/Software Development · Engineering - Telecom/Technology · DBA· Computer Science· Database · **SQL**· Programming· Configuration· Archiving

SQL Server DBA

Digi Visions - Cairo, Egypt

21 days ago

Full Time **Work From Home**

Experienced · 5 - 8 Yrs of Exp · IT/Software Development · Engineering - Telecom/Technology· Active Directory · Computer Science· Information Technology (IT)· Administration · MS **SQL** · MS **SQL** Server · **SQL** · **SQL** Server

Traditional Files

- If an application needs to store only a small amount of data, traditional files work well.
- Traditional files are not practical when a large amount of data must be stored and manipulated.
- Many businesses keep millions of data items.
- When a traditional file contains this much data, simple operations such as searching, inserting, and deleting become inefficient.
- When developing applications that work with large amounts of data, developers prefer to use a database instead of traditional files.

Database

- A database is a collection of related data.
- By data, we mean known facts that can be recorded and that have implicit meaning.
- For example, consider the names, telephone numbers, and addresses of the people you know.
- Data is useless until it is processed and organized.
- When data is processed, organized, structured or presented in a given context so as to make it useful, it is called information.

Database

- A database is a shared collection of related data, and a description of this data, designed to meet the information needs of an organization.

STUDENT

Name	Student_number	Class	Major
Smith	17	1	CS
Brown	8	2	CS

COURSE

Course_name	Course_number	Credit_hours	Department
Intro to Computer Science	CS1310	4	CS
Data Structures	CS3320	4	CS
Discrete Mathematics	MATH2410	3	MATH
Database	CS3380	3	CS

Database: Examples

- University Database
- Airlines Database
- Hypermarket Database
- GIS Databases
- Library Catalogues
- Medical Records
- Bank Accounts
- Stock Control
- Product Catalogues
- Customer Histories

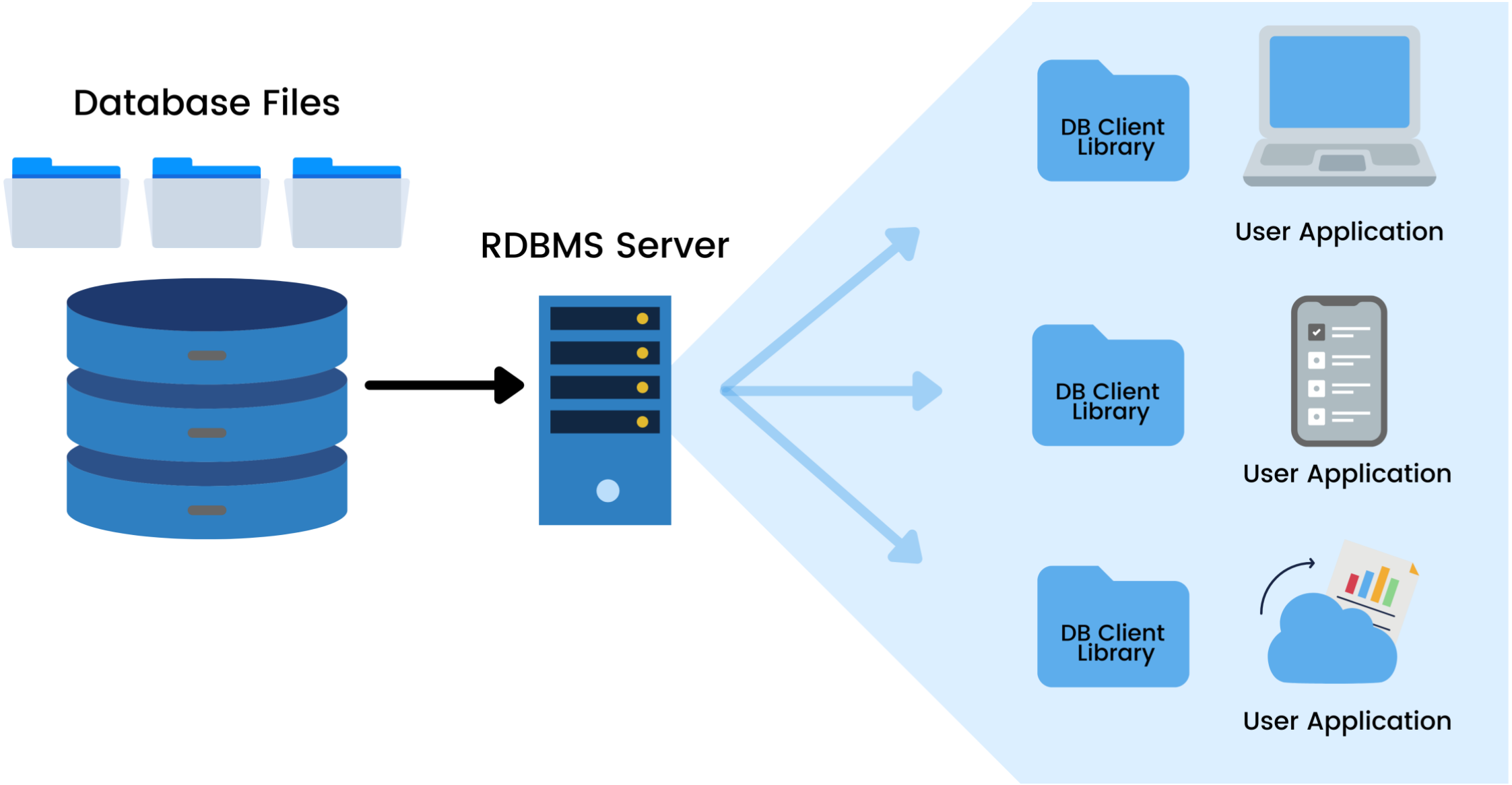
Database

- A database is a collection of related data.
- It mainly contains two main components:
  - Entities
  - Relationships
- Databases are not just a bunch of tables.
- A database also includes relationships between the different tables.

Database Management System (DBMS)

- A database management system (DBMS) is software that is specifically designed to store, retrieve, and manipulate large amounts of data in an organized and efficient manner.
  - SQLite
  - Oracle
  - Microsoft Access
  - MySQL
  - SQL Server

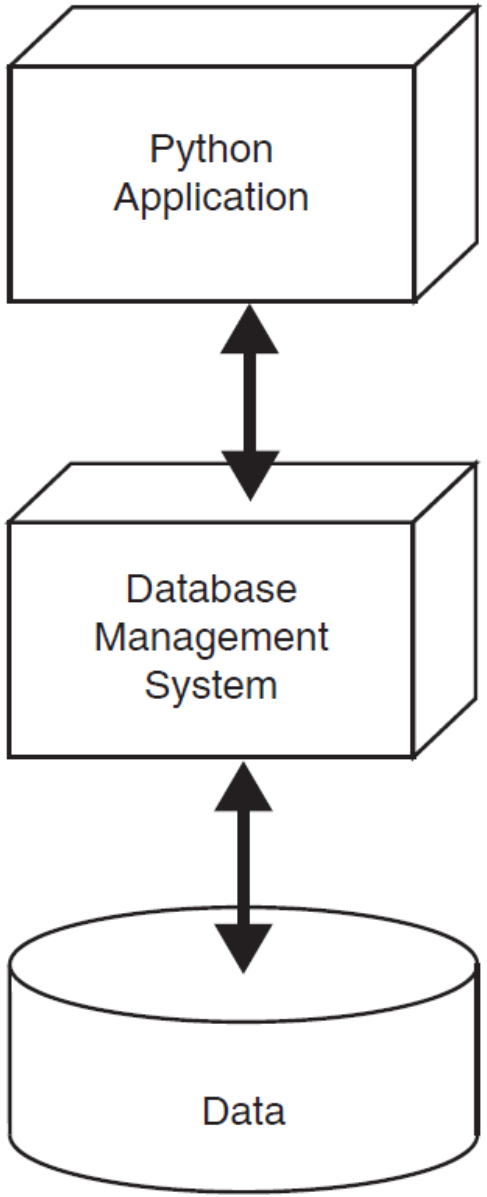
Database Management System (DBMS)



Database Management System (DBMS)

- An application that is developed in Python or another language can be written to use a DBMS to manage its data.
- Rather than retrieving or manipulating the data directly, the application can send instructions to the DBMS.
- The DBMS carries out those instructions and sends the results back to the application.
- The programmer needs to know only how to interact with the DBMS.

Database Management System (DBMS)



Database Management System (DBMS)

- Suppose a company keeps all its product records in a database.
- The company has a Python application that allows the user to look up information on any product by entering its product name.
- The Python application instructs the DBMS to retrieve the record for the product with the specified product name.
- The DBMS retrieves the product record and sends the data back to the Python application.
- The Python application displays the data to the user.

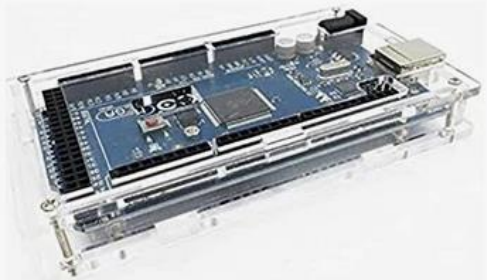
Database Management System (DBMS)

All | Arduino Mega 2560 | EN | Hello, sign in Account & Lists | Orders | 0 Cart

ones | Help | Electronics | Appliances | Prime | Fashion | Home | Video Games | FREE DELIVERY* on First Order | FREE Returns* | Cash on Delivery

ga 2560" | Sort by: Featured

RESULTS



Sponsored

Arduino Mega 2560 Acrylic Case

★☆☆☆☆ 1

EGP 99.99

Arduino Mega 2560 REV3 [A000067]

★★★★★ 1,251

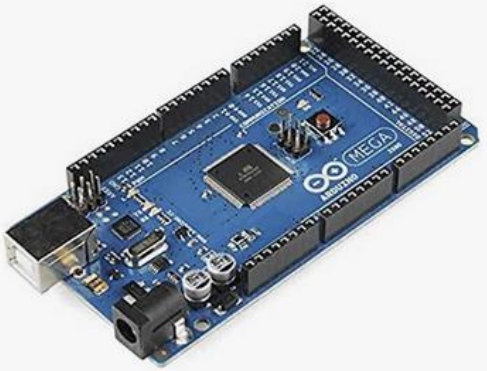
EGP 1,500.00

Get it **Saturday, February 18** -

Arduino Mega 2560 Compatible Base on Atmega2560 Mcu

★★★★★ 6

EGP 595.00



Arduino Mega 2560 R3 Microcontroller Board

★★★½☆ 14

EGP 575.00

Get it as soon as **Saturday,**

SQLite

- There are numerous DBMSs in use today, and Python can interact with many of them.
- Some of the more popular ones are MySQL, Oracle, Microsoft SQL Server, and SQLite.
- In this course, we use SQLite because it is easy to use.

SQL (Structured Query Language)

- SQL (Structured Query Language) is a standard language for working with the DBMS.
- SQL statements are submitted to the DBMS, and are instructions for the DBMS to carry out operations on data.
- Although SQL is a language, you don't use it to write applications.
- It is intended only as a standard means of interacting with a DBMS.
- You still need a general programming language, such as Python, to write an application for the ordinary user.

Tables, Rows, and Columns

- Database is a collection of related tables.

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
Alicia	J	Zelaya	999887777	1968-01-19	3321 Castle, Spring, TX	F	25000	987654321	4
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	NULL	1

DEPARTMENT

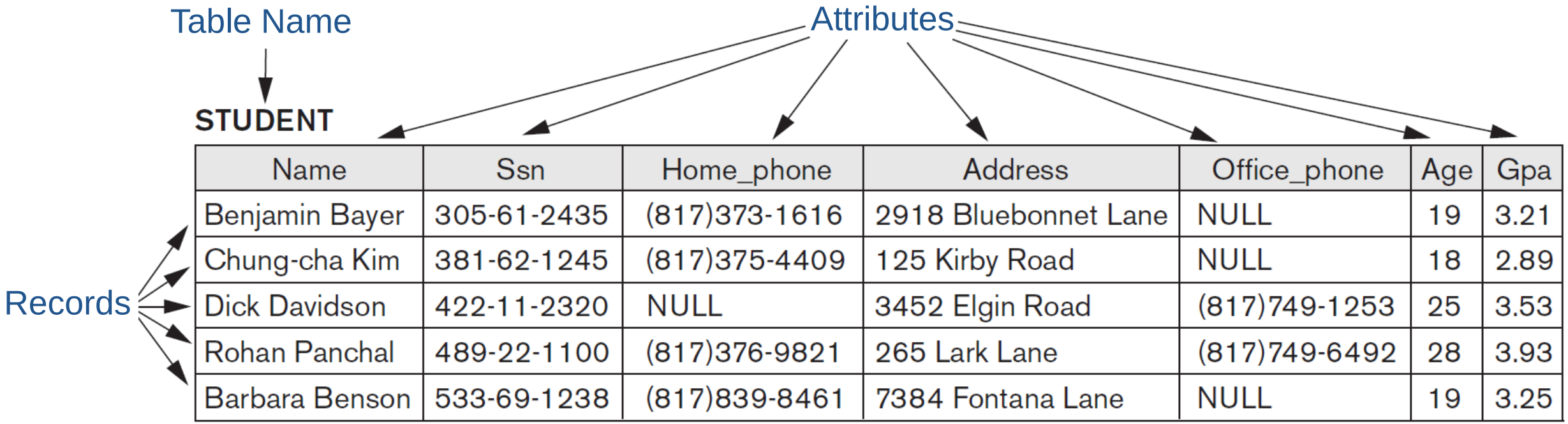
Dname	Dnumber	Mgr_ssn	Mgr_start_date
Research	5	333445555	1988-05-22
Administration	4	987654321	1995-01-01
Headquarters	1	888665555	1981-06-19

DEPT_LOCATIONS

Dnumber	Dlocation
1	Houston
4	Stafford
5	Bellaire
5	Sugarland
5	Houston

Tables, Rows, and Columns

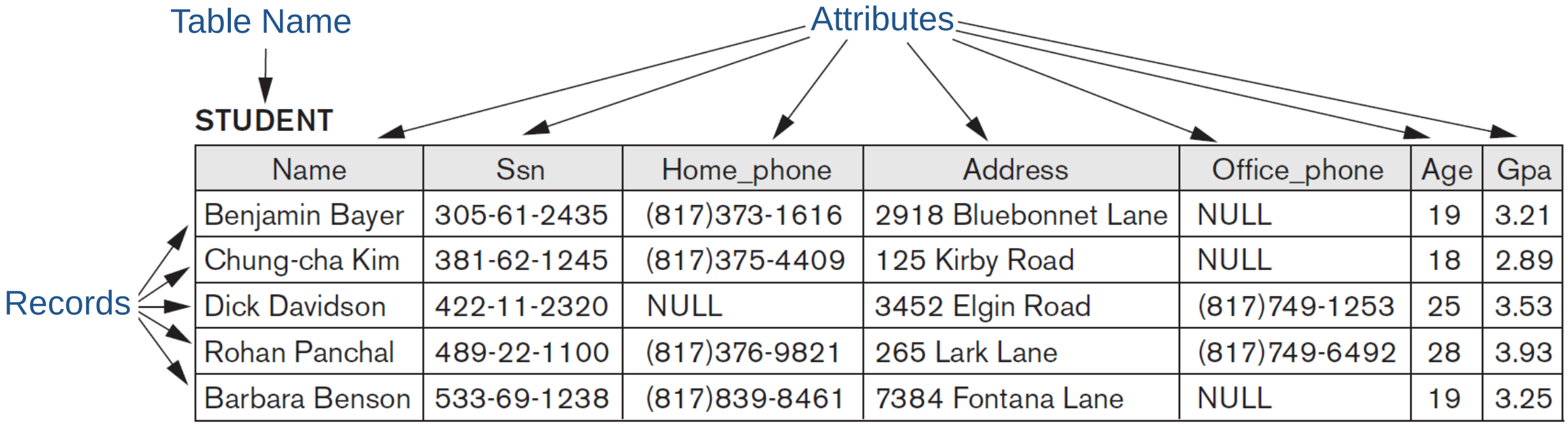
- Table is a collection of related records.



Name	Ssn	Home_phone	Address	Office_phone	Age	Gpa
Benjamin Bayer	305-61-2435	(817)373-1616	2918 Bluebonnet Lane	NULL	19	3.21
Chung-cha Kim	381-62-1245	(817)375-4409	125 Kirby Road	NULL	18	2.89
Dick Davidson	422-11-2320	NULL	3452 Elgin Road	(817)749-1253	25	3.53
Rohan Panchal	489-22-1100	(817)376-9821	265 Lark Lane	(817)749-6492	28	3.93
Barbara Benson	533-69-1238	(817)839-8461	7384 Fontana Lane	NULL	19	3.25

Tables, Rows, and Columns

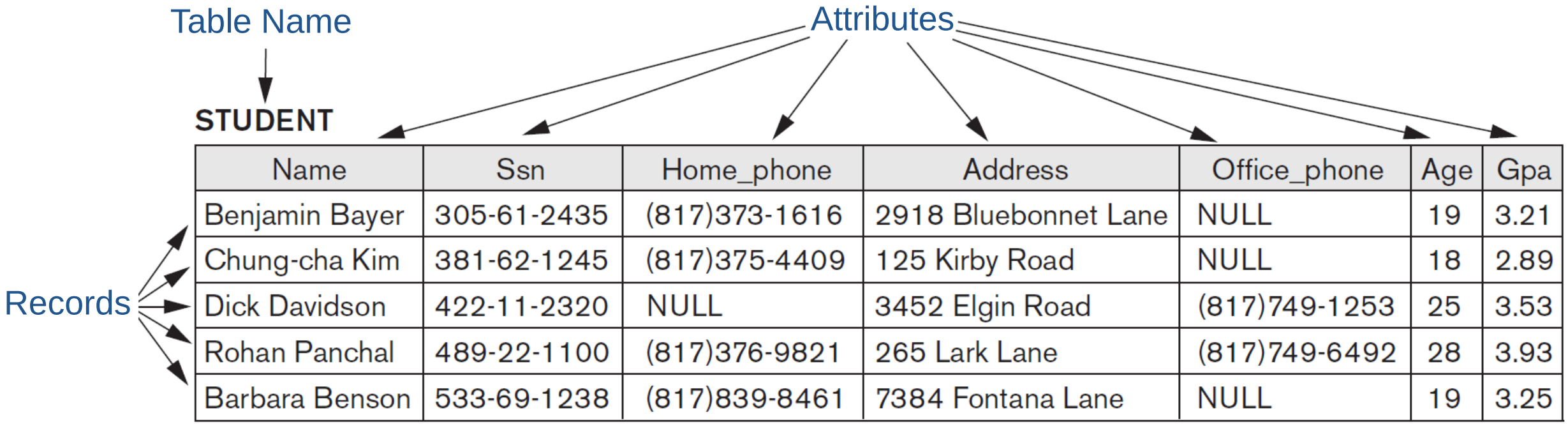
- Record is a collection of related fields.



Name	Ssn	Home_phone	Address	Office_phone	Age	Gpa
Benjamin Bayer	305-61-2435	(817)373-1616	2918 Bluebonnet Lane	NULL	19	3.21
Chung-cha Kim	381-62-1245	(817)375-4409	125 Kirby Road	NULL	18	2.89
Dick Davidson	422-11-2320	NULL	3452 Elgin Road	(817)749-1253	25	3.53
Rohan Panchal	489-22-1100	(817)376-9821	265 Lark Lane	(817)749-6492	28	3.93
Barbara Benson	533-69-1238	(817)839-8461	7384 Fontana Lane	NULL	19	3.25

Tables, Rows, and Columns

- Field is individual piece of data about the item, represents an attribute.



Name	Ssn	Home_phone	Address	Office_phone	Age	Gpa
Benjamin Bayer	305-61-2435	(817)373-1616	2918 Bluebonnet Lane	NULL	19	3.21
Chung-cha Kim	381-62-1245	(817)375-4409	125 Kirby Road	NULL	18	2.89
Dick Davidson	422-11-2320	NULL	3452 Elgin Road	(817)749-1253	25	3.53
Rohan Panchal	489-22-1100	(817)376-9821	265 Lark Lane	(817)749-6492	28	3.93
Barbara Benson	533-69-1238	(817)839-8461	7384 Fontana Lane	NULL	19	3.25

Column Data Types

- When you create a database table, you must specify a data type for the columns.

SQLite Data Type	Description
INTEGER	Integer number
REAL	Real number
TEXT	String
BLOB	Binary Large Object

Primary Keys

- Database tables usually have a primary key, which is a column that can be used to identify a specific row.
- The column that is designated as the primary key must hold a unique value each row.
- Here are some examples:
- A table stores employee data, and because each employee's ID number is unique, this column can be used as the primary key.
- A table stores product data, and because each product has a unique serial number, this column can be used as the primary key.

Primary Keys

- A primary key is a unique identifier of records in a table.

EMPLOYEE

Fname	Lname	Ssn	Bdate	Address	Sex	Salary	Dno
John	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	5
Franklin	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	5
Alicia	Zelaya	999887777	1968-01-19	3321 Castle, Spring, TX	F	25000	4
Jennifer	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	4
Ramesh	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	5
Joyce	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	5
Ahmad	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	4
James	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	1

Primary Keys

- The primary key value cannot be NULL.

EMPLOYEE

Fname	Lname	Ssn	Bdate	Address	Sex	Salary	Dno
John	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	5
Franklin	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	5
Alicia	Zelaya	999887777	1968-01-19	3321 Castle, Spring, TX	F	25000	4
Jennifer	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	4
Ramesh	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	5
Joyce	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	5
Ahmad	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	4
James	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	1

Primary Keys

- Primary key values may be generated manually or automatically.

PROJECT

Pname	Pnumber	Plocation	Dnum
ProductX	1	Bellaire	5
ProductY	2	Sugarland	5
ProductZ	3	Houston	5

Primary Keys

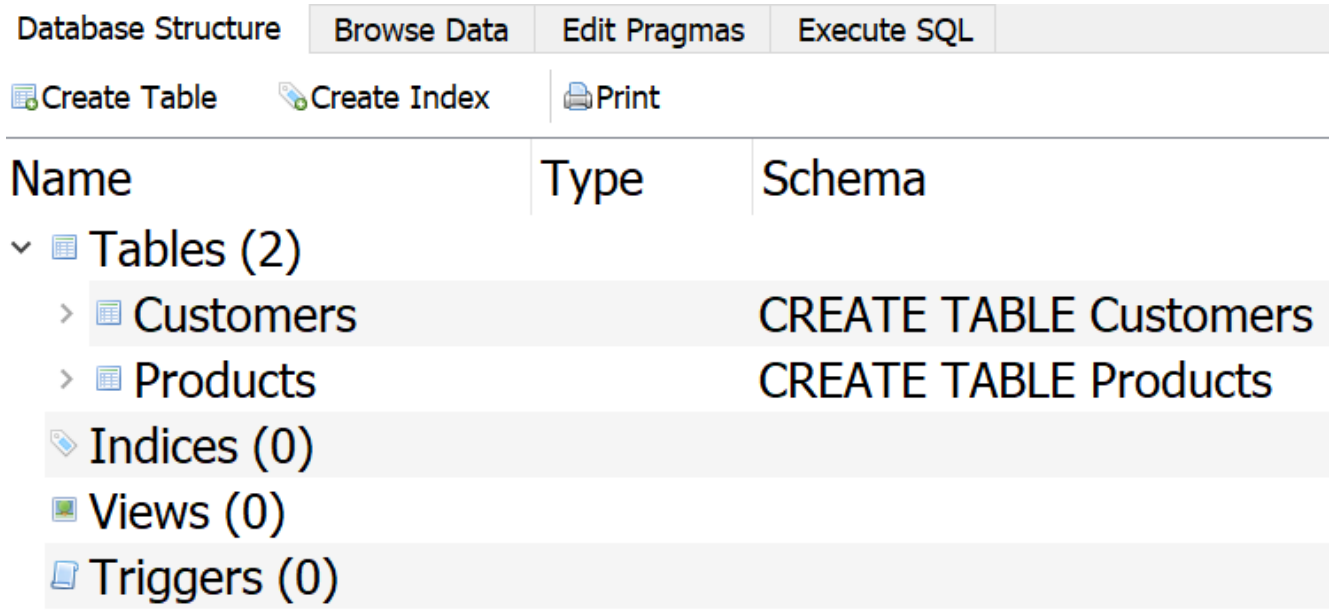
- Primary key can be one field or more.

WORKS_ON

Essn	Pno	Hours
123456789	1	32.5
123456789	2	7.5
666884444	3	40.0
453453453	1	20.0

Chocolate Database

- In our examples, we will work with a sample database named `chocolate.db`.
- The database contains data from a fictitious company that sells chocolate products.



Chocolate Database

- The database contains a `Products` table.

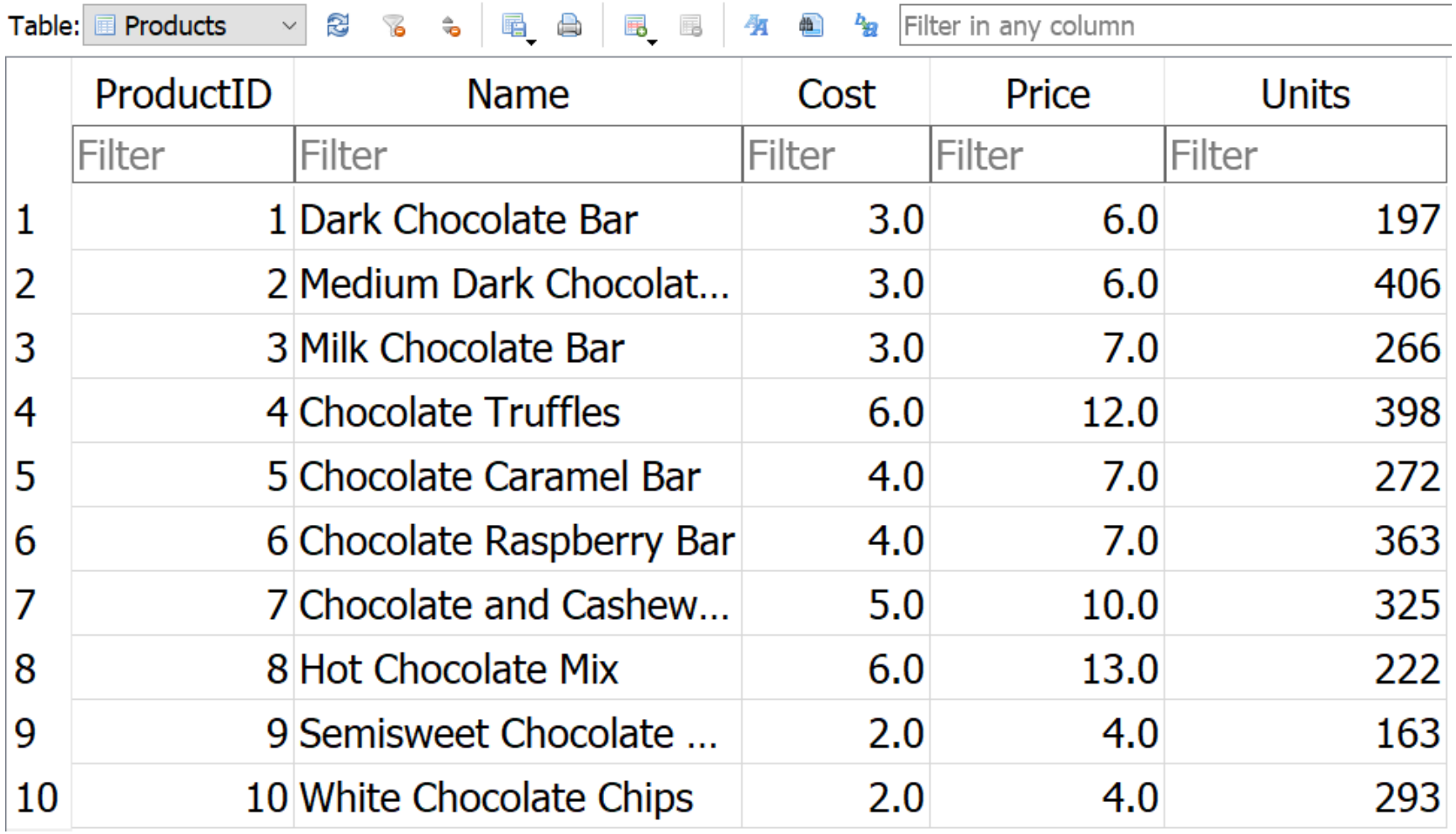


Table: Products

Filter in any column

	ProductID	Name	Cost	Price	Units
	Filter	Filter	Filter	Filter	Filter
1	1	Dark Chocolate Bar	3.0	6.0	197
2	2	Medium Dark Chocolat...	3.0	6.0	406
3	3	Milk Chocolate Bar	3.0	7.0	266
4	4	Chocolate Truffles	6.0	12.0	398
5	5	Chocolate Caramel Bar	4.0	7.0	272
6	6	Chocolate Raspberry Bar	4.0	7.0	363
7	7	Chocolate and Cashew...	5.0	10.0	325
8	8	Hot Chocolate Mix	6.0	13.0	222
9	9	Semisweet Chocolate ...	2.0	4.0	163
10	10	White Chocolate Chips	2.0	4.0	293

Chocolate Database

- The database contains a `Customers` table.

Table: Customers Filter in any column

	CustomerID	FirstName	LastName	StreetAddress	City	State	ZipCode	Phone
	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	1	Marianne	Romer	28 Texas Point	Schenectady	New York	12325	518-555-0060
2	2	Kip	Prandini	0497 Amoth Avenue	Lynn	Massachusetts	1905	781-555-5143
3	3	Vanni	Merida	0012 Everett Pass	Prescott	Arizona	86305	928-555-6758
4	4	Baldwin	Koba	25 Darwin Crossing	Lynchburg	Virginia	24503	434-555-4547
5	5	Jaquelyn	D'Orsay	650 Aberg Trail	Bethesda	Maryland	20816	240-555-0019
6	6	Elaina	Jaume	2707 Mockingbird Circle	Brooksville	Florida	34605	352-555-3810
7	7	Pincas	Phillpotts	884 Hayes Lane	Mount Vernon	New York	10557	914-555-9661
8	8	Wilhelmine	Meysham	1072 Manufacturers ...	Shawnee Mission	Kansas	66205	913-555-8815
9	9	Mariette	Dymidowicz	9 Oakridge Point	San Diego	California	92165	619-555-3307
10	10	Gisele	Klewi	65 Sommers Junction	Jacksonville	Florida	32259	904-555-0156

Querying Data

- The SELECT statement is used in SQL to retrieve data from a database.

CustomerID	FirstName	LastName	StreetAddress	City	State	ZipCode	Phone
1	Marianne	Romer	28 Texas Point	Schenectady	New York	12325	518-555-0060
2	Kip	Prandini	0497 Amoth Avenue	Lynn	Massachusetts	1905	781-555-5143
3	Vanni	Merida	0012 Everett Pass	Prescott	Arizona	86305	928-555-6758
4	Baldwin	Koba	25 Darwin Crossing	Lynchburg	Virginia	24503	434-555-4547
5	Jaquelyn	D'Orsay	650 Aberg Trail	Bethesda	Maryland	20816	240-555-0019
6	Elaina	Jaume	2707 Mockingbird Circle	Brooksville	Florida	34605	352-555-3810
7	Pincas	Phillpotts	884 Hayes Lane	Mount Vernon	New York	10557	914-555-9661
8	Wilhelmine	Meysham	1072 Manufacturers ...	Shawnee Mission	Kansas	66205	913-555-8815
9	Mariette	Dymidowicz	9 Oakridge Point	San Diego	California	92165	619-555-3307
10	Gisele	Klewi	65 Sommers Junction	Jacksonville	Florida	32259	904-555-0156

Querying Data: Selecting All the Columns

```
SELECT * FROM Customers
```

CustomerID	FirstName	LastName	StreetAddress	City	State	ZipCode	Phone
1	Marianne	Romer	28 Texas Point	Schenectady	New York	12325	518-555-0060
2	Kip	Prandini	0497 Amoth Avenue	Lynn	Massachusetts	1905	781-555-5143
3	Vanni	Merida	0012 Everett Pass	Prescott	Arizona	86305	928-555-6758
4	Baldwin	Koba	25 Darwin Crossing	Lynchburg	Virginia	24503	434-555-4547
5	Jaquelyn	D'Orsay	650 Aberg Trail	Bethesda	Maryland	20816	240-555-0019
6	Elaina	Jaume	2707 Mockingbird Circle	Brooksville	Florida	34605	352-555-3810
7	Pincas	Phillpotts	884 Hayes Lane	Mount Vernon	New York	10557	914-555-9661
8	Wilhelmine	Meysham	1072 Manufacturers ...	Shawnee Mission	Kansas	66205	913-555-8815
9	Mariette	Dymidowicz	9 Oakridge Point	San Diego	California	92165	619-555-3307
10	Gisele	Klewi	65 Sommers Junction	Jacksonville	Florida	32259	904-555-0156

```
Result: 50 rows returned in 8ms
```

Querying Data: Selecting All the Columns

```
SELECT * FROM Products
```

ProductID	Name	Cost	Price	Units
1	Dark Chocolate Bar	3.0	6.0	197
2	Medium Dark Chocolate Bar	3.0	6.0	406
3	Milk Chocolate Bar	3.0	7.0	266
4	Chocolate Truffles	6.0	12.0	398
5	Chocolate Caramel Bar	4.0	7.0	272
6	Chocolate Raspberry Bar	4.0	7.0	363
7	Chocolate and Cashew Bar	5.0	10.0	325
8	Hot Chocolate Mix	6.0	13.0	222
9	Semisweet Chocolate Chips	2.0	4.0	163
10	White Chocolate Chips	2.0	4.0	293

```
Result: 10 rows returned in 4ms
```

Querying Data: Selecting Specific Columns

```
SELECT Name FROM Products
```

Name
Dark Chocolate Bar
Medium Dark Chocolate Bar
Milk Chocolate Bar
Chocolate Truffles
Chocolate Caramel Bar
Chocolate Raspberry Bar
Chocolate and Cashew Bar
Hot Chocolate Mix
Semisweet Chocolate Chips
White Chocolate Chips

Querying Data: Selecting Specific Columns

```
SELECT Name, Cost FROM Products
```

Name	Cost
Dark Chocolate Bar	3.0
Medium Dark Chocolate Bar	3.0
Milk Chocolate Bar	3.0
Chocolate Truffles	6.0
Chocolate Caramel Bar	4.0
Chocolate Raspberry Bar	4.0
Chocolate and Cashew Bar	5.0
Hot Chocolate Mix	6.0
Semisweet Chocolate Chips	2.0
White Chocolate Chips	2.0

Querying Data: Selecting Specific Columns

```
SELECT ProductID, Name, Price, Units
FROM Products
```

ProductID	Name	Price	Units
1	Dark Chocolate Bar	6.0	197
2	Medium Dark Chocolate Bar	6.0	406
3	Milk Chocolate Bar	7.0	266
4	Chocolate Truffles	12.0	398
5	Chocolate Caramel Bar	7.0	272
6	Chocolate Raspberry Bar	7.0	363
7	Chocolate and Cashew Bar	10.0	325
8	Hot Chocolate Mix	13.0	222
9	Semisweet Chocolate Chips	4.0	163
10	White Chocolate Chips	4.0	293

Querying Data: SQL Relational Operators

Operator	Meaning
>	Greater-Than
<	Less-Than
>=	Greater-Than or Equal-To
<=	Less-Than or Equal-To
==	Equal-To
=	Equal-To
!=	Not Equal-To
<>	Not Equal-To

Querying Data: Specifying Search Criteria

```
SELECT * FROM Products
WHERE Price >= 10
```

ProductID	Name	Cost	Price	Units
4	Chocolate Truffles	6.0	12.0	398
7	Chocolate and Cashew Bar	5.0	10.0	325
8	Hot Chocolate Mix	6.0	13.0	222

Querying Data: Specifying Search Criteria

```
SELECT * FROM Products
WHERE Units < 200
```

ProductID	Name	Cost	Price	Units
1	Dark Chocolate Bar	3.0	6.0	197
9	Semisweet Chocolate ...	2.0	4.0	163

Querying Data: Specifying Search Criteria

```
SELECT * FROM Products
WHERE Price <> 6
```

ProductID	Name	Cost	Price	Units
3	Milk Chocolate Bar	3.0	7.0	266
4	Chocolate Truffles	6.0	12.0	398
5	Chocolate Caramel Bar	4.0	7.0	272
6	Chocolate Raspberry Bar	4.0	7.0	363
7	Chocolate and Cashew...	5.0	10.0	325
8	Hot Chocolate Mix	6.0	13.0	222
9	Semisweet Chocolate ...	2.0	4.0	163
10	White Chocolate Chips	2.0	4.0	293

Querying Data: SQL Logical Operators

- SQL Logical Operators: `AND`, `OR`, and `NOT`.
- You can use the `AND` and `OR` logical operators to specify multiple search criteria in a `WHERE` clause.

Querying Data: SQL Logical Operators

```
SELECT * FROM Products
WHERE Price > 5 AND Units > 300
```

ProductID	Name	Cost	Price	Units
2	Medium Dark Chocolat...	3.0	6.0	406
4	Chocolate Truffles	6.0	12.0	398
6	Chocolate Raspberry Bar	4.0	7.0	363
7	Chocolate and Cashew...	5.0	10.0	325

Querying Data: SQL Logical Operators

```
SELECT * FROM Products
WHERE Price >= 8 OR Units > 300
```

ProductID	Name	Cost	Price	Units
2	Medium Dark Chocolat…	3.0	6.0	406
4	Chocolate Truffles	6.0	12.0	398
6	Chocolate Raspberry Bar	4.0	7.0	363
7	Chocolate and Cashew…	5.0	10.0	325
8	Hot Chocolate Mix	6.0	13.0	222

Querying Data: String Comparisons

```
SELECT * FROM Products
WHERE Name = "Hot Chocolate Mix"
```

ProductID	Name	Cost	Price	Units
8	Hot Chocolate Mix	6.0	13.0	222

Querying Data: String Comparisons

```
SELECT * FROM Products
WHERE lower(Name) = "hot chocolate mix"
```

ProductID	Name	Cost	Price	Units
8	Hot Chocolate Mix	6.0	13.0	222

Querying Data: Using the LIKE Operator

```
SELECT * FROM Products
WHERE Name LIKE "%dark%"
```

ProductID	Name	Cost	Price	Units
1	Dark Chocolate Bar	3.0	6.0	197
2	Medium Dark Chocolat...	3.0	6.0	406

Querying Data: Using the LIKE Operator

```
SELECT * FROM Products
WHERE Name LIKE "dark%"
```

ProductID	Name	Cost	Price	Units
1	Dark Chocolate Bar	3.0	6.0	197

Querying Data: Using the LIKE Operator

```
SELECT * FROM Products
WHERE Name LIKE "%bar"
```

ProductID	Name	Cost	Price	Units
1	Dark Chocolate Bar	3.0	6.0	197
2	Medium Dark Chocolate Bar	3.0	6.0	406
3	Milk Chocolate Bar	3.0	7.0	266
5	Chocolate Caramel Bar	4.0	7.0	272
6	Chocolate Raspberry Bar	4.0	7.0	363
7	Chocolate and Cashew Bar	5.0	10.0	325

Querying Data: The Concatenation Operator

```
SELECT FirstName || LastName
FROM Customers
```

FirstName || LastName
MarianneRomer
KipPrandini
VanniMerida
BaldwinKoba
JaquelynD'Orsay
ElainaJaume
PincasPhillpotts
WilhelmineMeysham

Querying Data: The Concatenation Operator

```
SELECT FirstName || " " || LastName
FROM Customers
```

stName || " " || LastNan
Marianne Romer
Kip Prandini
Vanni Merida
Baldwin Koba
Jaquelyn D'Orsay
Elaina Jaume
Pincas Phillpotts
Wilhelmine Meysham

Querying Data: ALIAS Syntax

```
SELECT FirstName || " " || LastName AS "Full Name"
FROM Customers
```

Full Name
Marianne Romer
Kip Prandini
Vanni Merida
Baldwin Koba
Jaquelyn D'Orsay
Elaina Jaume
Pincas Phillpotts
Wilhelmine Meysham

Querying Data: Sorting the Results

```
SELECT * FROM Products
ORDER BY Price
```

ProductID	Name	Cost	Price	Units
9	Semisweet Chocolate ...	2.0	4.0	163
10	White Chocolate Chips	2.0	4.0	293
1	Dark Chocolate Bar	3.0	6.0	197
2	Medium Dark Chocolat...	3.0	6.0	406
3	Milk Chocolate Bar	3.0	7.0	266
5	Chocolate Caramel Bar	4.0	7.0	272
6	Chocolate Raspberry Bar	4.0	7.0	363
7	Chocolate and Cashew...	5.0	10.0	325
4	Chocolate Truffles	6.0	12.0	398
8	Hot Chocolate Mix	6.0	13.0	222

Querying Data: Sorting the Results

```
SELECT * FROM Products
ORDER BY Price ASC
```

ProductID	Name	Cost	Price	Units
9	Semisweet Chocolate ...	2.0	4.0	163
10	White Chocolate Chips	2.0	4.0	293
1	Dark Chocolate Bar	3.0	6.0	197
2	Medium Dark Chocolat...	3.0	6.0	406
3	Milk Chocolate Bar	3.0	7.0	266
5	Chocolate Caramel Bar	4.0	7.0	272
6	Chocolate Raspberry Bar	4.0	7.0	363
7	Chocolate and Cashew...	5.0	10.0	325
4	Chocolate Truffles	6.0	12.0	398
8	Hot Chocolate Mix	6.0	13.0	222

Querying Data: Sorting the Results

```
SELECT * FROM Products
ORDER BY Price DESC
```

ProductID	Name	Cost	Price	Units
8	Hot Chocolate Mix	6.0	13.0	222
4	Chocolate Truffles	6.0	12.0	398
7	Chocolate and Cashew...	5.0	10.0	325
3	Milk Chocolate Bar	3.0	7.0	266
5	Chocolate Caramel Bar	4.0	7.0	272
6	Chocolate Raspberry Bar	4.0	7.0	363
1	Dark Chocolate Bar	3.0	6.0	197
2	Medium Dark Chocolat...	3.0	6.0	406
9	Semisweet Chocolate ...	2.0	4.0	163
10	White Chocolate Chips	2.0	4.0	293

Querying Data: Aggregate Functions

```
SELECT MAX(Price)
FROM Products
```

MAX(Price)
13.0

Querying Data: Aggregate Functions

```
SELECT MIN(Price)
FROM Products
```

MIN(Price)
4.0

Querying Data: Aggregate Functions

```
SELECT SUM(Price)
FROM Products
```

SUM(Price)
76.0

Querying Data: Aggregate Functions

```
SELECT AVG(Price)
FROM Products
```

AVG(Price)
7.6

Querying Data: Aggregate Functions

```
SELECT COUNT(*)
FROM Products
```

COUNT(*)
10

Querying Data: Aggregate Functions

```
SELECT COUNT(*)
FROM Products
WHERE Price > 5
```

COUNT(*)
8

Updating Rows

- You use the UPDATE statement in SQL to change the value of an existing row.
- Here is the general format of the UPDATE statement:

```
UPDATE Table
SET Column = Value
WHERE Criteria
```

Updating Rows

```
UPDATE Products
SET Name = "Kit Kat"
WHERE ProductID = 3
```

Table: Products

	ProductID	Name	Cost	Price	Units
	Filter	Filter	Filter	Filter	Filter
1	1	Dark Chocolate Bar	3.0	6.0	197
2	2	Medium Dark Chocolat...	3.0	6.0	406
3	3	Kit Kat	3.0	7.0	266
4	4	Chocolate Truffles	6.0	12.0	398
5	5	Chocolate Caramel Bar	4.0	7.0	272

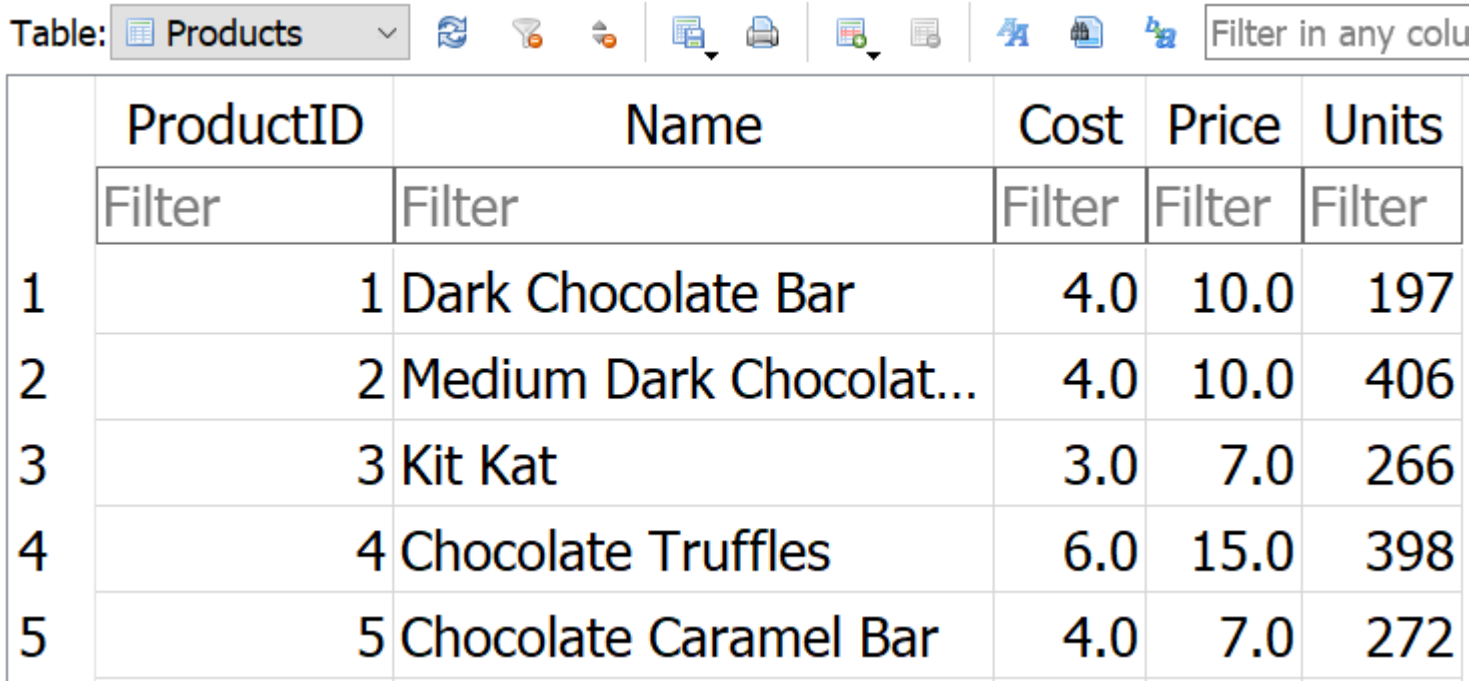
Updating Rows

```
UPDATE Products
SET Price = 15
WHERE Price >= 10
```

4	4	Chocolate Truffles	6.0	15.0	398
5	5	Chocolate Caramel Bar	4.0	7.0	272
6	6	Chocolate Raspberry Bar	4.0	7.0	363
7	7	Chocolate and Cashew...	5.0	15.0	325
8	8	Hot Chocolate Mix	6.0	15.0	222
9	9	Semisweet Chocolate ...	2.0	4.0	163
10	10	White Chocolate Chips	2.0	4.0	293

Updating Rows

```
UPDATE Products
SET Price = 10, Cost = 4
WHERE Price = 6
```



	ProductID	Name	Cost	Price	Units
	Filter	Filter	Filter	Filter	Filter
1	1	Dark Chocolate Bar	4.0	10.0	197
2	2	Medium Dark Chocolat...	4.0	10.0	406
3	3	Kit Kat	3.0	7.0	266
4	4	Chocolate Truffles	6.0	15.0	398
5	5	Chocolate Caramel Bar	4.0	7.0	272

Updating Rows: WARNING!

- Be careful that you do not leave out the `WHERE` clause and the conditional expression when using an `UPDATE` statement.
- You could change the contents of every row in the table!
- For example, look at the following statement:

```
UPDATE Products
SET Price = 10
```

- Because this statement does not have a `WHERE` clause, it will change the `Price` column for every row in the `Products` table to 10!

Deleting Rows

- In SQL we use the `DELETE` statement to delete one or more rows from a table.
- The general format of the `DELETE` statement is:

```
DELETE FROM Table
WHERE Criteria
```

Deleting Rows

```
DELETE FROM Products
WHERE ProductID = 10
```

	ProductID	Name	Cost	Price	Units
	Filter	Filter	Filter	Filter	Filter
1	1	Dark Chocolate Bar	4.0	10.0	197
2	2	Medium Dark Chocolat...	4.0	10.0	406
3	3	Kit Kat	3.0	7.0	266
4	4	Chocolate Truffles	6.0	15.0	398
5	5	Chocolate Caramel Bar	4.0	7.0	272
6	6	Chocolate Raspberry Bar	4.0	7.0	363
7	7	Chocolate and Cashew...	5.0	15.0	325
8	8	Hot Chocolate Mix	6.0	15.0	222
9	9	Semisweet Chocolate ...	2.0	4.0	163

Deleting Rows

```
DELETE FROM Products
WHERE Price = 7
```

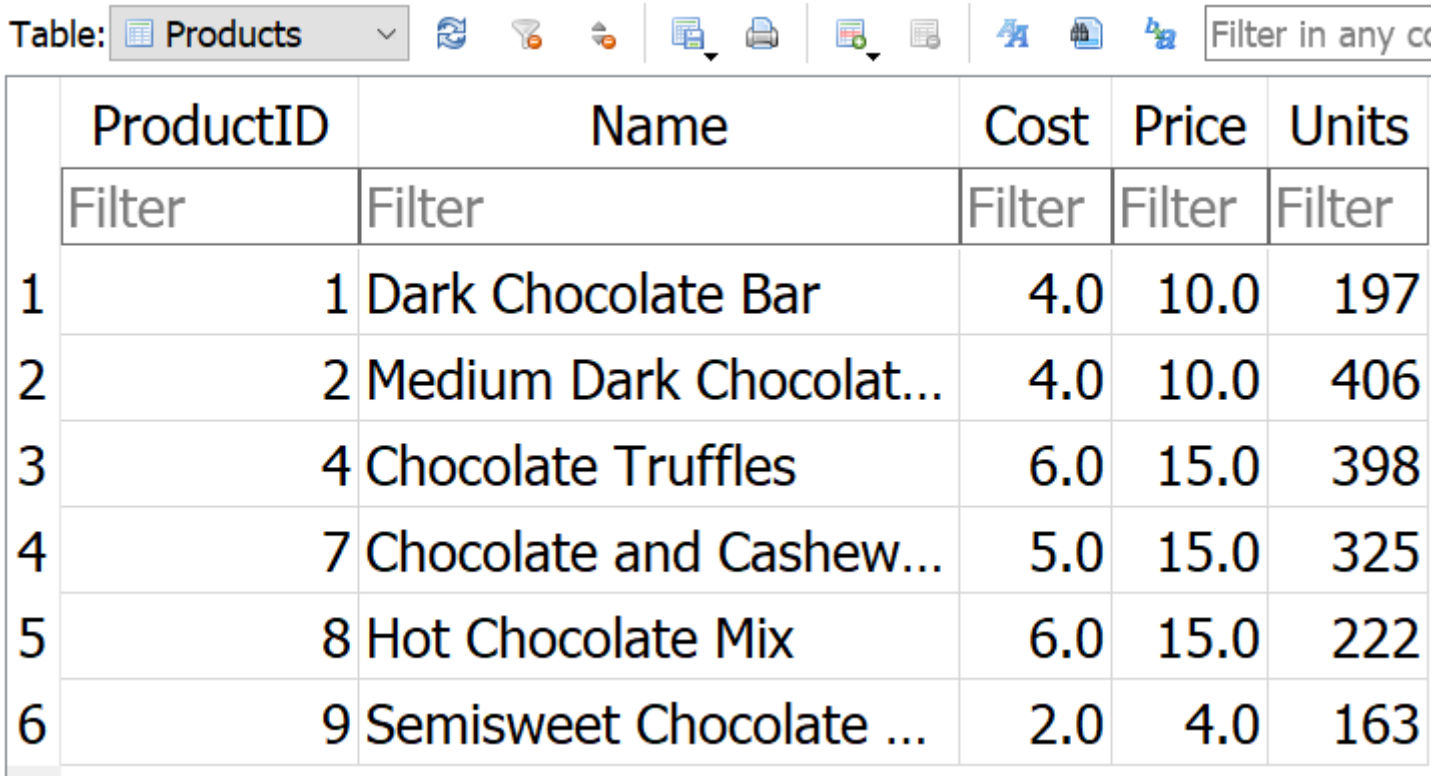


Table: Products

Filter in any co

	ProductID	Name	Cost	Price	Units
	Filter	Filter	Filter	Filter	Filter
1	1	Dark Chocolate Bar	4.0	10.0	197
2	2	Medium Dark Chocolat...	4.0	10.0	406
3	4	Chocolate Truffles	6.0	15.0	398
4	7	Chocolate and Cashew...	5.0	15.0	325
5	8	Hot Chocolate Mix	6.0	15.0	222
6	9	Semisweet Chocolate ...	2.0	4.0	163

Deleting Rows: WARNING!

- Be careful that you do not leave out the `WHERE` clause and the conditional expression when using a `DELETE` statement.
- You could delete every row in the table!
- For example, look at the following statement:

```
DELETE FROM Products
```

- Because this statement does not have a `WHERE` clause, it will delete every row in the `Products` table!

Adding Data to a Table

- We use the `INSERT` statement in SQL to insert a new row into a table.
- Once you have created a database file, and created one or more tables in the database, you can add rows to the table(s).
- Here is the general format:

```
INSERT INTO TableName (ColName1, ColName2, etc...)
VALUES (Value1, Value2, etc...)
```

Adding Data to a Table

```
INSERT INTO Products (Name, Cost, Price, Units)
VALUES ("Moro", 7, 15, 100)
```

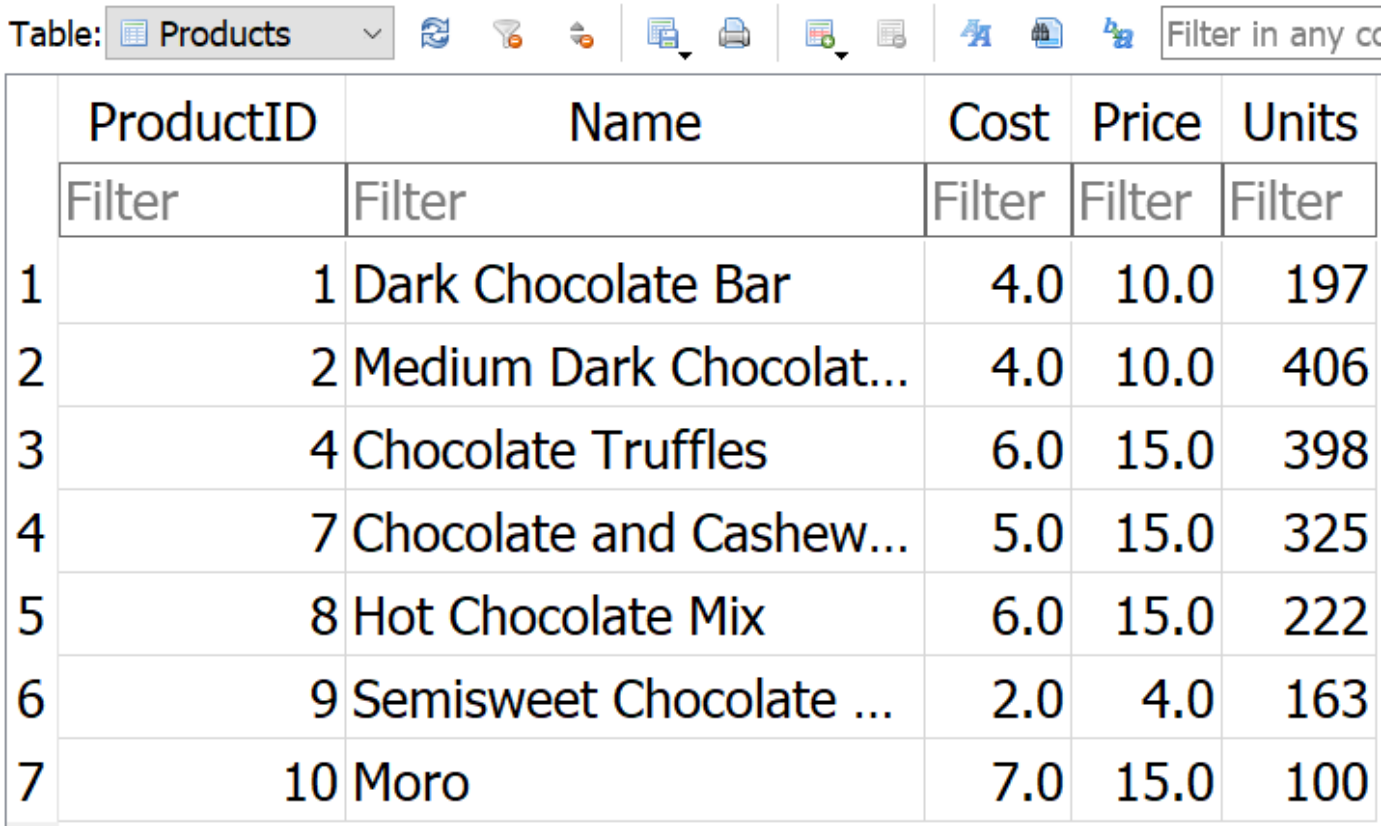


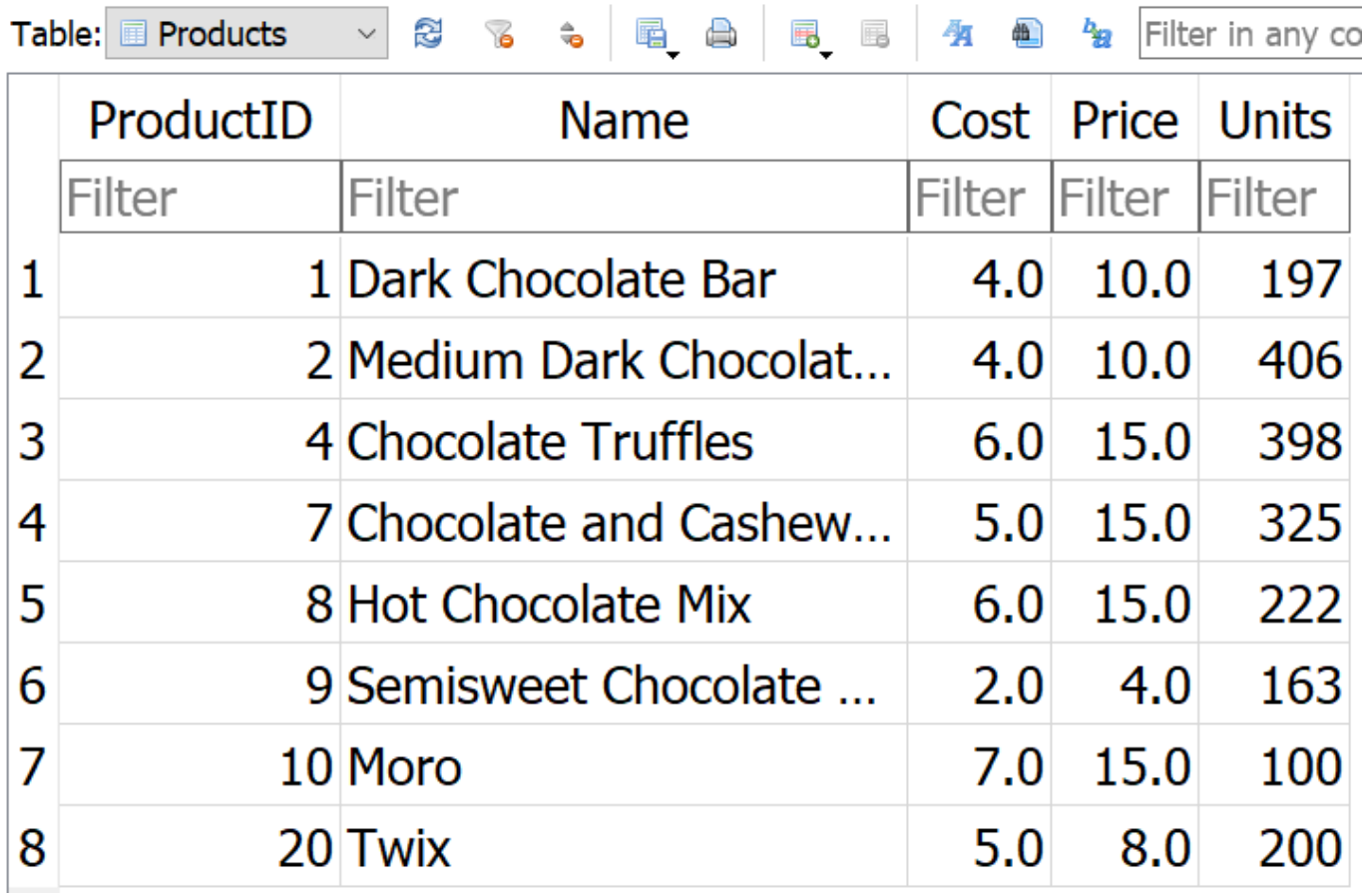
Table: Products

Filter in any co

	ProductID	Name	Cost	Price	Units
	Filter	Filter	Filter	Filter	Filter
1	1	Dark Chocolate Bar	4.0	10.0	197
2	2	Medium Dark Chocolat...	4.0	10.0	406
3	4	Chocolate Truffles	6.0	15.0	398
4	7	Chocolate and Cashew...	5.0	15.0	325
5	8	Hot Chocolate Mix	6.0	15.0	222
6	9	Semisweet Chocolate ...	2.0	4.0	163
7	10	Moro	7.0	15.0	100

Adding Data to a Table

```
INSERT INTO Products (ProductID, Name, Cost, Price, Units)
VALUES (20, "Twix", 5, 8, 200)
```



	ProductID	Name	Cost	Price	Units
	Filter	Filter	Filter	Filter	Filter
1	1	Dark Chocolate Bar	4.0	10.0	197
2	2	Medium Dark Chocolat...	4.0	10.0	406
3	4	Chocolate Truffles	6.0	15.0	398
4	7	Chocolate and Cashew...	5.0	15.0	325
5	8	Hot Chocolate Mix	6.0	15.0	222
6	9	Semisweet Chocolate ...	2.0	4.0	163
7	10	Moro	7.0	15.0	100
8	20	Twix	5.0	8.0	200

Company Database: Creating Tables

```
CREATE Table Employees
(
      EmpID INTEGER PRIMARY KEY NOT NULL,
      Name TEXT,
      Salary REAL,
      Position TEXT
)
```

Company Database: Adding Data to a Table

```
INSERT INTO Employees (Name, Salary, Position)
VALUES  ("Kareem Ramy", 15000, "Manager");

INSERT INTO Employees (Name, Salary, Position)
VALUES  ("Ahmed Ali", 7000, "Engineer");

INSERT INTO Employees (Name, Salary, Position)
VALUES  ("Omar Alaa", 6000, "Designer");

INSERT INTO Employees (Name, Salary, Position)
VALUES  ("Ahmed Gamal", 6000, "Designer");

INSERT INTO Employees (Name, Salary, Position)
VALUES  ("Mohamed Samy", 6000, "Designer");
```

Company Database: Querying the Database

```
SELECT * FROM Employees
```

EmpID	Name	Salary	Position
1	Kareem Ramy	15000.0	Manager
2	Ahmed Ali	7000.0	Engineer
3	Omar Alaa	6000.0	Designer
4	Ahmed Gamal	6000.0	Designer
5	Mohamed Samy	6000.0	Designer

Company Database: Creating Tables

```
CREATE TABLE Departments
(
      DeptID INTEGER PRIMARY KEY NOT NULL,
      Name TEXT
)
```

Company Database: Adding Data to a Table

```
INSERT INTO Departments (Name)
VALUES ("R&D");

INSERT INTO Departments (Name)
VALUES ("Marketing");

INSERT INTO Departments (Name)
VALUES ("Accounting");
```

Company Database: Querying the Database

```
SELECT * FROM Departments
```

	DeptID	Name
1	1	R&D
2	2	Marketing
3	3	Accounting

Deleting a Table

- If you need to delete a table, you can use the SQL statement `DROP TABLE`.
- Here is the general format of the `DROP TABLE` statement:

```
DROP TABLE TableName
```

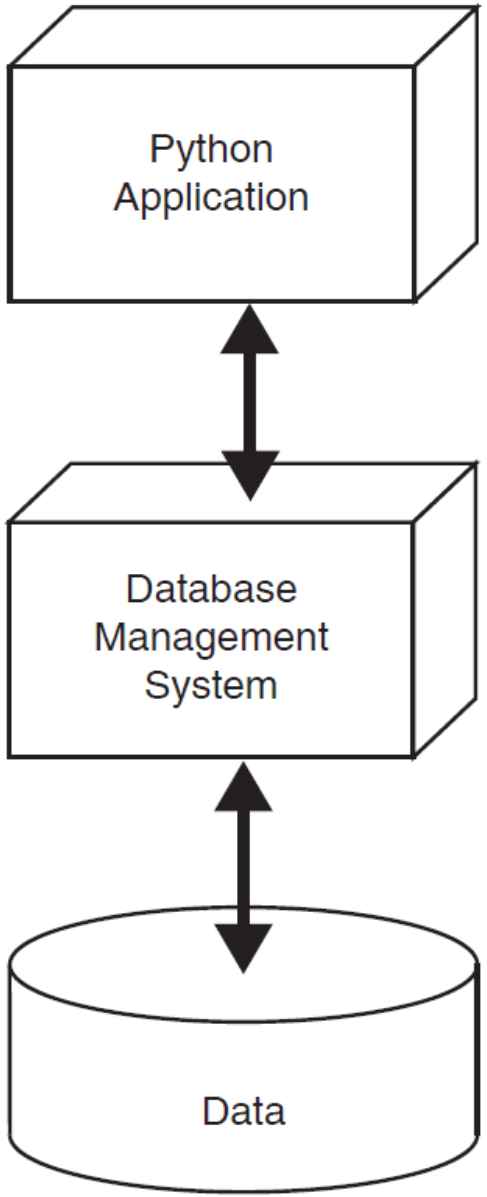
- For example,

```
DROP TABLE Departments
```

CRUD Operations

- The four basic operations of a database application are Create, Read, Update, and Delete.
- CRUD stands for Create, Read, Update, and Delete.
  - **Create:** The process of creating new set of data in the database.
  - **Read:** The process of reading an existing set of data from the database.
  - **Update:** The process of changing, or updating, an existing set of data in the database
  - **Delete:** The process of deleting a set of data from the database.

Database Management System (DBMS)



Company Database: Querying Data With Python

```
import sqlite3

# Connect to the database.
conn = sqlite3.connect('Company.db')

# Get a cursor.
cur = conn.cursor()

# Get the data of all employees.
cur.execute('SELECT * FROM Employees')

# Fetch the results of the query.
results = cur.fetchall()

# Iterate over the rows and display the results.
for row in results:
    print(row)

# Close the database connection.
conn.close()
```

Company Database: Querying Data With Python

```
# Output
(1, 'Kareem Ramy', 15000.0, 'Manager')
(2, 'Ahmed Ali', 7000.0, 'Engineer')
(3, 'Omar Alaa', 6000.0, 'Designer')
(4, 'Ahmed Gamal', 6000.0, 'Designer')
(5, 'Mohamed Samy', 6000.0, 'Designer')
```

Company Database: Querying Data With Python

```
import sqlite3

# Connect to the database.
conn = sqlite3.connect('Company.db')

# Get a cursor.
cur = conn.cursor()

# Get the data of all employees.
cur.execute('SELECT * FROM Employees')

# Fetch the results of the query.
results = cur.fetchall()

# Iterate over the rows and display the results.
for row in results:
    for col in row:
        print(col, end='\t')
    print()

# Close the database connection.
conn.close()
```

Company Database: Querying Data With Python

```
# Output
1       Kareem Ramy     15000.0 Manager
2       Ahmed Ali       7000.0  Engineer
3       Omar Alaa       6000.0  Designer
4       Ahmed Gamal     6000.0  Designer
5       Mohamed Samy    6000.0  Designer
```

Company Database: Adding Data to a Table With Python

```
import sqlite3

# Get the data from the user.
name = input('Enter Name: ')
salary  = float(input('Enter Salary: '))
position = input('Enter Position: ')

# Connect to the database.
conn = sqlite3.connect('Company.db')

# Get a cursor.
cur = conn.cursor()

# Add a row to the Employees table.
cur.execute('INSERT INTO Employees (Name, Salary, Position) VALUES (?, ?, ?)', (name, salary, position))

# Commit the changes.
conn.commit()

# Close the database connection.
conn.close()
```

Company Database: Searching Data With Python

```
import sqlite3

# Get the id from the user.
emp_id = int(input('Enter ID: '))

# Connect to the database.
conn = sqlite3.connect('Company.db')

# Get a cursor.
cur = conn.cursor()

# Send the SELECT statement to the DBMS.
cur.execute('SELECT * FROM Employees WHERE EmpId = ?', (emp_id,))

# Fetch the results of the query.
results = cur.fetchall()

# Iterate over the rows and display the results.
for row in results:
    print(row)

# Close the database connection.
conn.close()
```

Company Database: Searching Data With Python

```
import sqlite3

# Get the name from the user.
name = input('Enter Name: ')

# Connect to the database.
conn = sqlite3.connect('Company.db')

# Get a cursor.
cur = conn.cursor()

# Send the SELECT statement to the DBMS.
cur.execute('SELECT * FROM Employees WHERE Name LIKE ?', (f'%{name}%',))

# Fetch the results of the query.
results = cur.fetchall()

# Iterate over the rows and display the results.
for row in results:
    print(row)

# Close the database connection.
conn.close()
```

Company Database: Deleting Data With Python

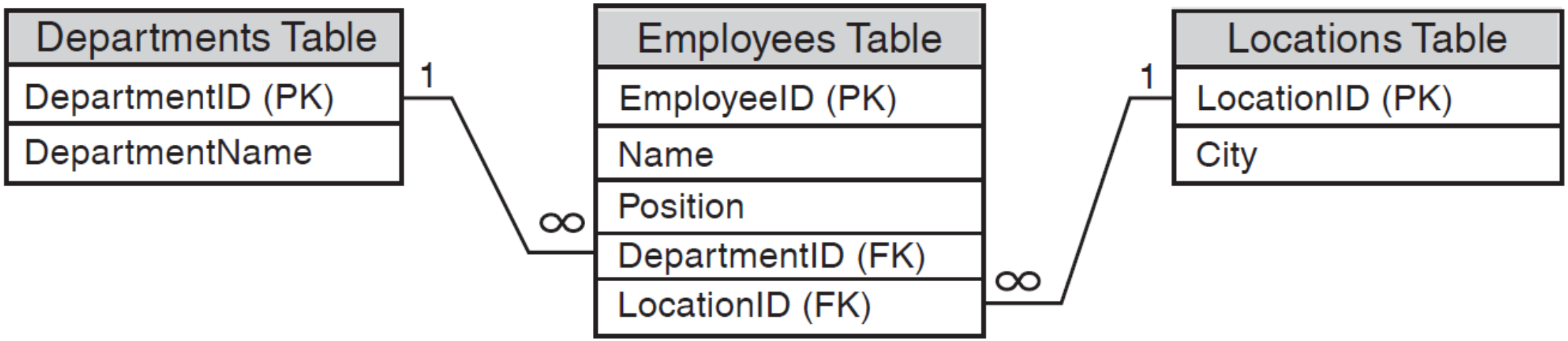
```
import sqlite3

# Get the id from the user.
emp_id = int(input('Enter ID: '))

# Connect to the database.
conn = sqlite3.connect('Company.db')

# Get a cursor.
cur = conn.cursor()

# Delete a row from the Employees table.
cur.execute('DELETE FROM Employees WHERE EmpId = ?', (emp_id,))

# Commit the changes.
conn.commit()

# Close the database connection.
conn.close()
```

Company Database: Updating Data With Python

```
import sqlite3

# Get the data from the user.
emp_id = input('Enter ID: ')
name = input('Enter Name: ')
salary  = float(input('Enter Salary: '))
position = input('Enter Position: ')

# Connect to the database.
conn = sqlite3.connect('Company.db')

# Get a cursor.
cur = conn.cursor()

# Update a row in the Employees table.
cur.execute('UPDATE Employees SET Name = ?, Salary = ?, Position = ? WHERE EmpId = ?', (name,
salary, position, emp_id))

# Commit the changes.
conn.commit()

# Close the database connection.
conn.close()
```

Relational Data

- In a relational database, a column from one table can be associated with a column from other tables.
- This association creates a relationship between the tables.



Relational Data

Employees Table

EmployeeID	Name	Position	DepartmentID	LocationID
1	Arlene Meyers	Director	4	4
2	Janelle Grant	Engineer	2	1
3	Jack Smith	Manager	3	3
4	Sonia Alvarado	Auditor	1	2
5	Renee Kincaid	Designer	3	3
6	Curt Green	Supervisor	2	1
7	Angela Taylor	Programmer	4	4

Departments Table

DepartmentID	DepartmentName
1	Accounting
2	Manufacturing
3	Marketing
4	Research and Development

Locations Table

LocationID	City
1	Austin
2	Boston
3	New York City
4	San Jose