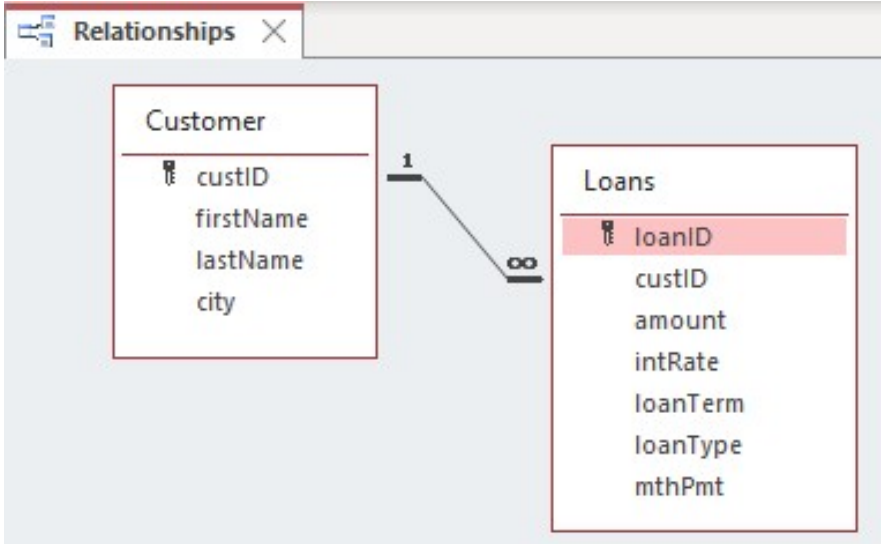
Database Programming

Outline

- An Introduction to Databases
  - Relational databases
  - Database management systems
    - Download and install SQLite and SQLite Studio
- Database Programming
  - Designing a simple DB application
    - Querying a single table
  - More complex DB applications
    - Querying multiple related tables
    - Using summary queries

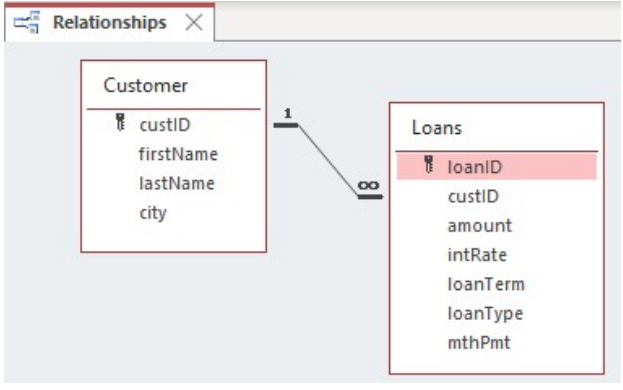
Introduction to Databases

- Databases everywhere
  - Airlines, hospitals, banks
- **Database** is a collection of related tables
  - Customer related to loans
- **Table** is a rectangular array of data
- Each column called a **field**
  - custID, firstName, lastName, …
- Each row called a **record**
  - Different information of the same type for each of the customers



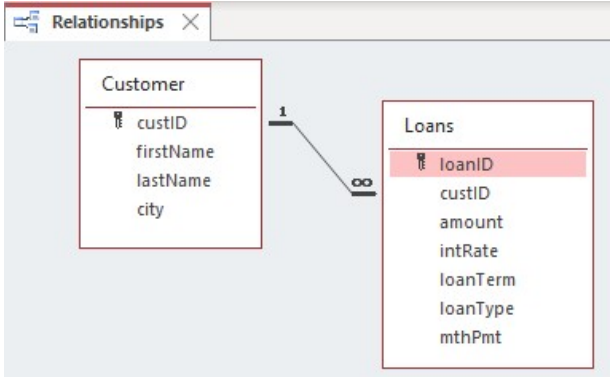
Primary and Foreign Keys

- **Primary key** – a field used to uniquely identify each record in a database table
  - **custID** in **Customer** table is the primary key
- **Foreign key** – a field in a table whose values must match the primary key values in a related table
  - **custID** in **Loans** table is the foreign key
  - A loan must be associated with a single customer
    - A single **Loans.custID** value in the **Loans** table must match one and only one **Customer.custID** value in the **Customer** table
  - A customer can be associated with multiple loans
    - A single value of **Customer.custID** from the **Customer** table can appear multiple times in **Loans.custID** column in the **Loans** table



Joining Two Tables

- **Relationship** between two tables
  - **custID** is a common field explicitly defining the relationship between the **Customer** and **Loans** tables
- Foreign keys allow us to **join** two (or more) tables in a relational database
  - **Customer** table is joined with **Loans** table
    - **Loans.custID** field in the **Loans** table is joined with **Customer.custID** field in the **Customer** table



Database Management Software

- Database management software (DBMS)
  - A collection of software tools that manage relational databases
    - Creating tables, designing queries, forms, reports
    - Managing user access, fine-tuning performance
  - Some of the prominent DBMSs
    - Enterprise level: Oracle, SQL Server, MySQL, AWS RDS, …
    - Desktop: **SQLite**, Microsoft Access, FileMaker
  - Python can communicate with any of those DBMSs
  - Python and other programming languages used to design application programs that interact with relational databases

Databases – Install SQLite

- Download and install SQLite

  https://github.com/pawelsalawa/sqlitestudio/releases

  – Download the `InstallSQLiteStudio` ZIP file (Windows) or DMG file (Mac)

  – Run the installation following the instructions

- Additional sites for SQLite and SQLiteStudio

  – SQLite

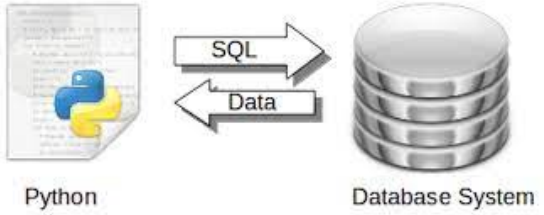
    - https://www.sqlitetutorial.net/download-install-sqlite/
    - https://www.sqlite.org/download.html

  – SQLiteStudio

    - https://sqlitestudio.pl/

- Use to test the SQL code used in Python programs

Database Programming



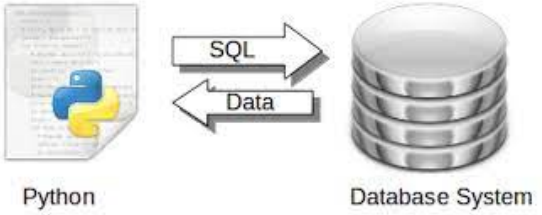
- **Lect14_DB_Program.py**
- `import sqlite3`
- **Connect** to the database
  - SQLite DB is just a file (**Loans.db**)

    ```
    db_conn = sqlite3.connect('Loans.db')
    ```

  - Typically a connection to a remote RDBMS (Oracle, MySQL, …)
- Get a **cursor** for the database
  - DB object used to access and manipulate the data in DB

    ```
    db_cursor = db_conn.cursor()
    ```

Database Programming (cont.)



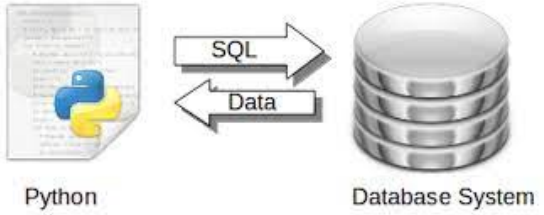
- Querying a single table (basic SQL):
  - Lists all records (rows) and fields (columns)

```
SELECT * FROM Customer
```

- Perform operations on the database
  - We are going to be using cursor only for retrieving the data out of DB
  - Could also insert, modify and delete records in the DB (see textbook)
    - `db_cursor.execute('SELECT * FROM Customer')`

	custID	firstName	lastName	city
1	100	Eileen	Cooper	Santa Fe
2	101	Jack	Smith	Santa Fe
3	102	Ryan	Murphy	Santa Fe
4	103	Scott	Hunter	Taos
5	104	Ellen	Harper	Albuquerque
6	105	Bob	Williams	Albuquerque
7	106	Max	Entermann	Santa Fe
8	107	Craig	Holden	Santa Fe
9	108	Helen	Rayus	Albuquerque
10	109	Ted	Myerson	Santa Fe
11	110	Peter	Sanger	Albuquerque
12	111	Rose	Budnick	Taos
13	112	Robert	Thompson	Taos
14	113	Austin	Powers	Albuquerque
15	114	Barbara	Ringer	Taos
16	115	Joseph	Rogers	Taos

Database Programming



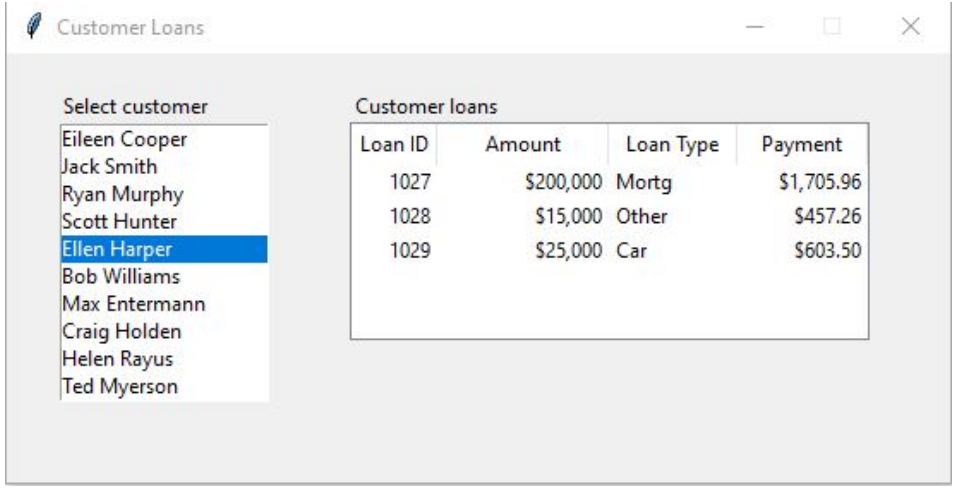
- Fetch and display the results
  - Use **fetchall()** method of the cursor object
    - `db_cursor.fetchall()`
  - Returns a list of tuples representing the resulting records

```
[(100, 'Eileen', 'Cooper', 'Santa Fe'),
 (101, 'Jack', 'Smith', 'Santa Fe'),
…
 (115, 'Joseph', 'Rogers', 'Taos')]
```

- Close the connection to the DB
  - Use the **close()** method of the connection object
    - `db_conn.close()`

Designing Loans DB Application

- Main program: **Lect14_Cust_Loans.py**
  - Create and size the main **tkinter** window
  - Call the customer **listbox** designer and return their IDs
  - Call the **ttk.Treeview** aka **table** widget setup
    - ttk.Treeview table widget will hold the resulting customer loans



  - Bind **<<ListboxSelect>>** event to functions that will implement the retrieval of loans for the selected customer

Listbox of Customer Names

- Select customer ID, first and last name

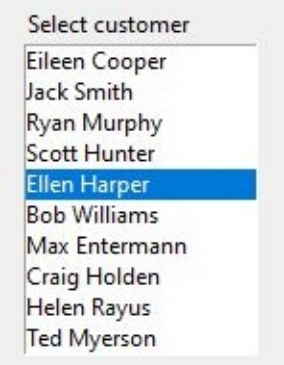
```
SELECT custID, firstName, lastName
FROM Customer
```

- Loop through the list of tuples

```
[(100, 'Eileen', 'Cooper'),
 (101, 'Jack', 'Smith'),
…
 (115, 'Joseph', 'Rogers')]
```

  - Build a list of customer IDs: `[100, 101, …, 115]`
  - Concatenate first and last names, **insert** into listbox
  - Return the list of customer IDs to main

	custID	firstName	lastName
1	100	Eileen	Cooper
2	101	Jack	Smith
3	102	Ryan	Murphy
4	103	Scott	Hunter
5	104	Ellen	Harper
6	105	Bob	Williams
7	106	Max	Entermann
8	107	Craig	Holden
9	108	Helen	Rayus
10	109	Ted	Myerson
11	110	Peter	Sanger
12	111	Rose	Budnick
13	112	Robert	Thompson
14	113	Austin	Powers
15	114	Barbara	Ringer
16	115	Joseph	Rogers



Treeview Table of Customer Loans

- Create a tuple of internal column IDs

```
column_ids = ('l_id','amt','l_type','pmt')
```

- Use them to define headings
  - **Loan ID**, **Amount**, **Loan Type** and **Payment**
- Configure positioning of values and column widths
  - Numbers are right aligned with **`anchor='e'`**
  - Widths adjusted by trial and error
  - Currency formatting performed in callback function later

Customer loans

Loan ID	Amount	Loan Type	Payment
1027	$200,000	Mortg	$1,705.96
1028	$15,000	Other	$457.26
1029	$25,000	Car	$603.50

<<ListboxSelect>> Callback Function

- Back in the main
  - Must deal with the default customer before the selection is changed to a different one
    - Get default customer's list index and use it to get their customer ID
    - Send the customer ID to **get_cust_loans** function to retrieve and display the default customer's loans
  - Bind **<<ListboxSelect>>** event to callback function
    - **get_cust_id** callback function called every time the user changes the selection, similar to repetitive clicks on a command button
    - The function gets customer's list-index and uses it to get their customer ID from the list of customer IDs passed to it
    - Calls **get_cust_loans** to retrieve and display the customer's loans

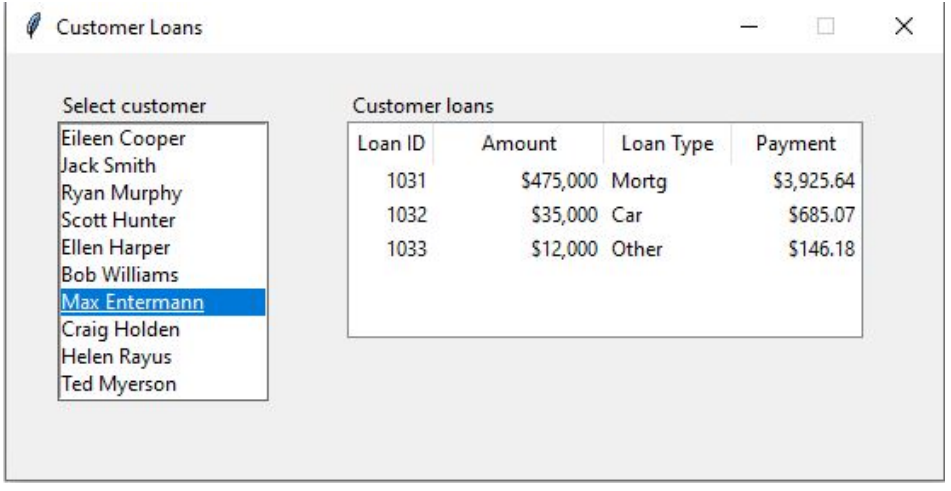
Retrieving Customer Loans

- **get_cust_loans** SQL statement
  - List selected loan information for a single customer

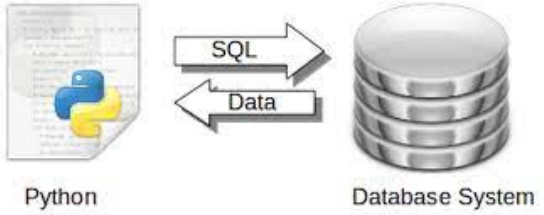
```
SELECT loanID, amount, loanType, mthPmt
FROM Loan WHERE custID=106
```

	loanID	amount	loanType	mthPmt
1	1031	475000	Mortg	3925.64
2	1032	35000	Car	685.07
3	1033	12000	Other	146.18

- Clear the table from previous customer loans
- Loop through loans of the current customer
  - List of tuples with 4 elements:
    - loan id, amount
    - loan type, monthly payment
  - Format amount and monthly payment as currency
  - Recreate the loan record tuple
  - Insert the formatted tuple into the Treeview table



Querying Multiple Tables



- Retrieving info from two tables:
  - List customer names, amount and monthly payment for mortgage loans from Santa Fe

```
SELECT firstName, lastName, amount, mthPmt
FROM Customer INNER JOIN Loan
  ON Customer.custID = Loan.custID
WHERE city = 'Santa Fe' AND loanType = 'Mortg'
```

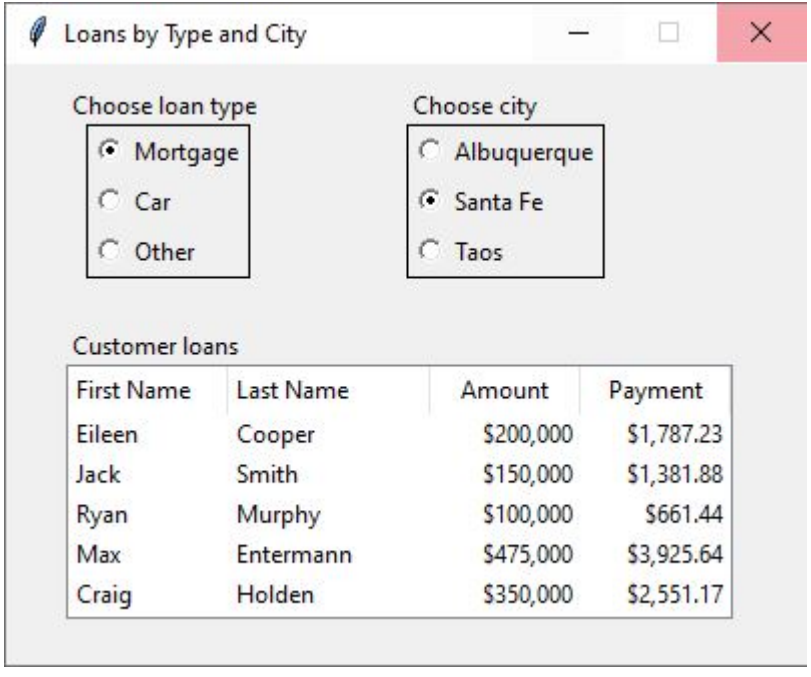
- Loop through resulting list of tuples:
  - Find the total amount of mortgage loans
  - Calculate the average monthly payment

	firstName	lastName	amount	mthPmt
1	Eileen	Cooper	200000	1787.23
2	Jack	Smith	150000	1381.88
3	Ryan	Murphy	100000	661.44
4	Max	Entermann	475000	3925.64
5	Craig	Holden	350000	2551.17
6	Ted	Myerson	525000	3300.48

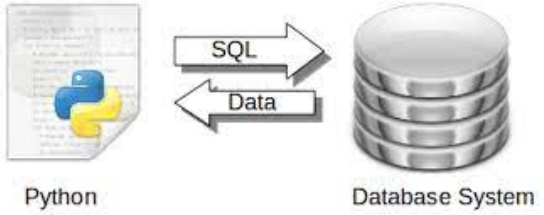
- **Lect14_DB_Program.py**

Loans by Type and City

- **Lect14_Loan_Type_City.py**
  - **Radiobuttons** used to set **StringVar()** objects to loan types and cities
  - **get_loan_types_city** callback function attached to each Radiobutton via **command** property
  - Gets the loan type and city values as **StringVar()** objects
  - Creates dynamic SQL string based on those values



Using Summary Queries



- Use SQL to summarize data:
  - Find the number of loans by loan type

```
SELECT loanType, COUNT(*) AS NumLoans
FROM Loan
GROUP BY loanType
```

	loanType	NumLoans
1	Car	7
2	Mortg	12
3	Other	6

  - Calculate the average mortgage monthly payment by city

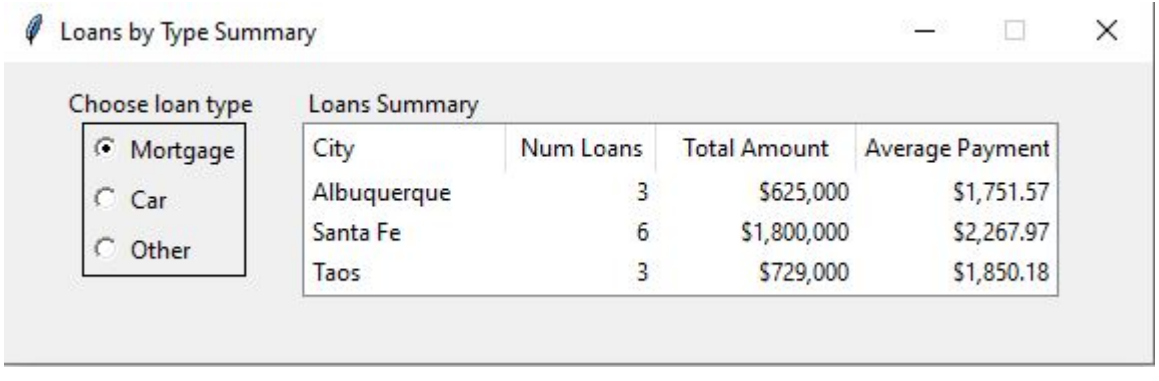
```
SELECT city, ROUND(AVG(mthPmt),2) AS avgMthPmt
FROM Customer INNER JOIN Loan
  ON Customer.custID = Loan.custID
WHERE loanType = 'Mortg'
GROUP BY city
```

	city	avgMthPmt
1	Albuquerque	1751.57
2	Santa Fe	2267.97
3	Taos	1850.18

- **Lect14_DB_Program.py**

Loans by Type Summary

- **Lect14_Loan_Type_Sum.py**
  - **Radiobuttons** used to set **StringVar()** objects to loan types
  - **get_loans_type_sum** callback function attached to each Radiobutton via **command** property
  - Gets the loan type as **StringVar()** objects
  - Creates dynamic summary SQL string based on the loan type



Summary

- Introduced relational database concepts
  - Tables, records, fields, queries, joins
  - SQLite: a simple RDBMS
- Python database programming
  - Import sqlite3, connect to database, get the cursor
  - Execute query using cursor, fetch all records as a list of tuples
  - Loop through the list of tuples, process and/or display
- Database application development
  - Using Listbox, Radiobuttons and Treeview widgets for DB GUI
  - Querying single or multiple tables and using summary queries