وزارة التربية
Ministry of Education
State of Kuwait | دولـــة الكويـت

تقنية المعلومات 11

الصـف الحادي عشر

الفصل الدراسي الأول

المرحلة الثانوية

وزارة التربية
Ministry of Education
State of Kuwait | دولـــة الكويـت

تقنية المعلومات 11

إشراف

منى سالم عوض سالم (رئيس اللجنة)

تأليف

رأفـت صابـر عبـداللاه أحمـــد
حســـــام الدين علي عبـد القادر

أشرف رضوان رضـوان سليمـان
د. يوسف منصور يوسف الخليفي

منى مرزوق مخلد العـــــازمي
مـنار مصطفى عبدالحميد جمال

إبراهيم عبدالله إبراهيم المياس

تصميم

ســـاره ياســـين عبدالله الأمـير

إخراج

أشرف رضوان رضـوان سليمـان

الطبعة الأولى
1447 هـ
2026/2025 م

الطبعة الأولى 2026/2025م

المراجعة العلمية

أشرف رضـــوان رضـــوان سليمـــان

فـــاطمة نجـــم جـــاسم الهـــولي

عبدالرحمن محمد مال الله الجزاف

المراجعة اللغوية

عـــبد الرازق محـــمد عصـــفور

الفريق المُساند

تصميم

سنيـــه محـــمد علـي المؤمـن

أودع بمكتبة الوزارة تحت رقم (55) بتاريخ 2025/9/2م


بسم الله الرحمن الرحيم

حضرة صاحب السمو الشيخ مشعل الأحمد الجابر الصباح

أمير دولة الكويت

H.H. Sheikh Meshal AL-Ahmad Al-Jaber Al-Sabah
Amir Of The State Of Kuwait

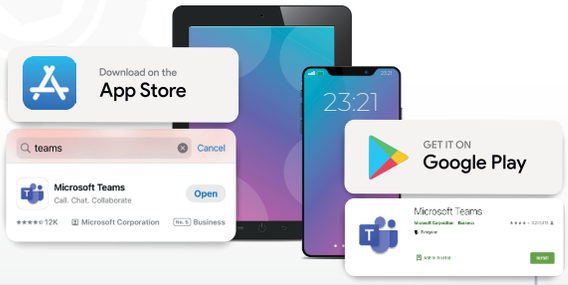
سمو الشيخ صباح خالد الحمد الصباح
ولي عهد دولة الكويت

H. H. Sheikh Sabah Khaled Al-Hamad Al-Sabah
Crown Prince Of The State Of Kuwait

الفصل الرقـــمي Digital Classroom

أولًا :تحميل وتثبيت تطبيق Microsoft Teams

الدخول على متجر تطبيقات الأجهزة الرقمية.



من الأجهزة الذكية

من جهاز الحاسوب

احرص على تحديث تطبيق Microsoft Teams بشكل دوري.

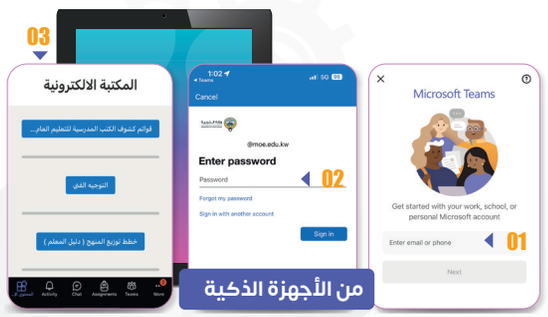
ثانياً: تفعيل Microsoft Teams على الجهاز الرقمي

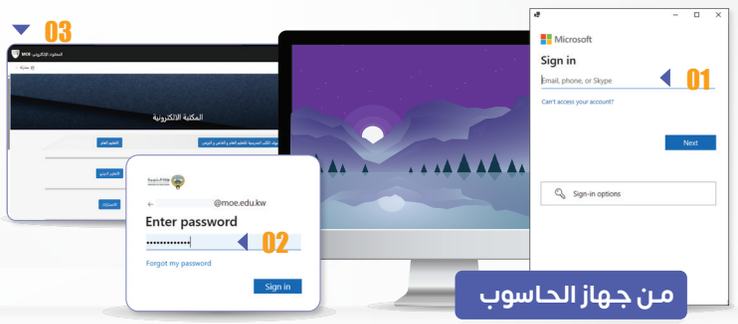
لتشغيل التطبيق اتبع الخطوات التالية:

01 أدخل اسم المستخدم: البريد الإلكتروني الرسمي الخاص بحساب Teams.

02 أدخل كلمة المرور الخاصة بحساب Teams.

03 تنقل بين واجهات التطبيق مثل المحتوى الإلكتروني، وفرق المواد الدراسية.





احتفظ بكلمة المرور في مكان آمن لتسهيل تسجيل الدخول لاحقًا.

لا تشارك بياناتك مع أي شخص غير موثوق به.

مـن هـو حـمد ؟...

هـو مُساعد تعليمـي رقمـي ذكـي يعتمـد علـى تقنيـات الـذكاء الاصطناعـي، مُصمـم لتقديـم الدعـم التعليمـي التفاعلـي لكافـة المناهـج الدراسـية.

حمد دائمًا معك... لتتعلّم بثقة وتتجح بامتياز.

مع حـمد Chat

مـا هـي خـدمـة حـمـد شـات؟

هـي خدمـة المحادثـة الذكيـة المقدمـة مـن وزارة التربيـة فـي دولـة الكويـت، تعتمـد علـى الـذكاء الاصطـناعي، تتمثـل وظيفتـه الأساسـية فـي تسـهيل عمليـة الفهـم والاسـتيعاب مـن خلال تقديـم الشـروحات المبسـطة، والإجابـة علـى الاستفسـارات، وتوجيـه المتعلميـن خلال مسـيرتهم التعليميـة.

أيـــن أجـــده؟

01

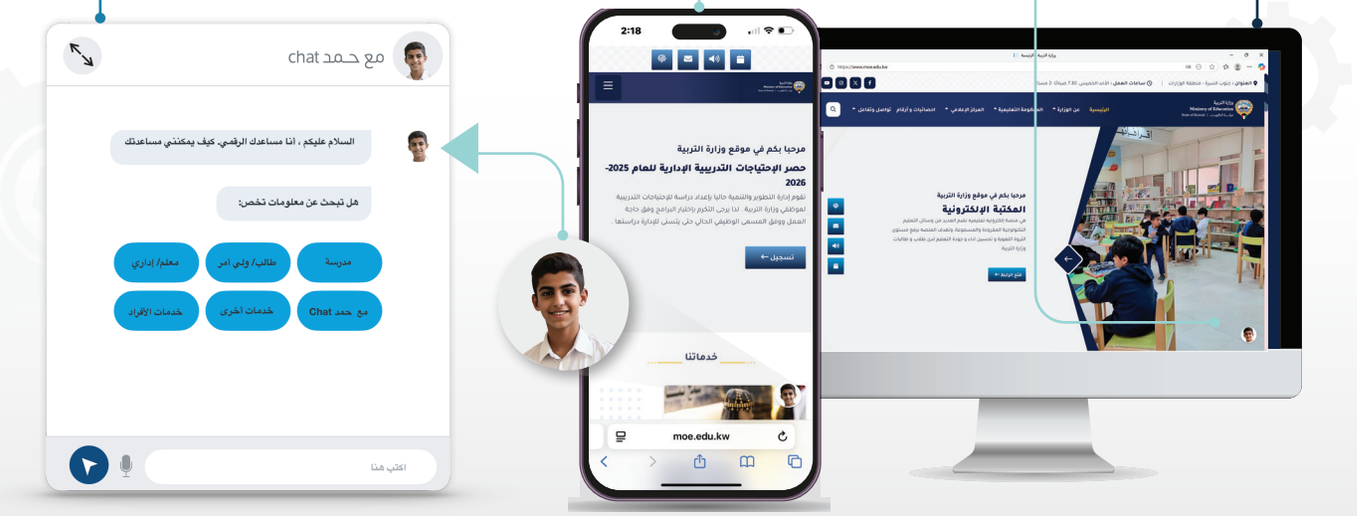
- زيارة الموقع الرسمي
www.moe.edu.Kw
- أو تطبيق وزارة التربية

02

الضغط عـلى
صورة حمد شات

03

اكتب استفساراتك، وستتلقى
الإجابات بشكل فوري.



المحـــتوى

العنوان	الصفحة

المقدمة

في عصر يشهد انفجارًا غير مسبوق في حجم البيانات وتنوع مصادرها، أصبحت القدرة على تنظيم هذه البيانات وتحليلها مهارة ضرورية في مختلف التخصصات والمجالات. ومن هنا تتضح أهمية قواعد البيانات كأداة مركزية تمكّن الأنظمة الرقمية من تخزين المعلومات واسترجاعها بكفاءة، وضمان سلامتها، وتحقيق الترابط المنطقي بينها.

يتناول هذا الكتاب مدخلاً علميًا مبسطًا لتعلم أساسيات قواعد البيانات باستخدام مُحرك SQLite، الذي يتميز ببساطته، سرعته، وسهولة دمجه في التطبيقات المكتوبة بلغة البرمجة Python، وهي من أكثر لغات البرمجة شيوعًا في العالم اليوم بفضل بُنيتها الواضحة ودعمها الواسع لمكتبات متعددة الأغراض.

يتم تناول الجانب العملي من خلال الربط البرمجي بين Python وSQLite باستخدام مكتبة sqlite3، حيث سنتعلم كيفية:

- إنشاء قواعد البيانات وتنفيذ الاستعلامات.
- حماية البيانات وتفادي مشكلات مثل هجمات الحقن البرمجي SQL Injection.

ومع التقدم المتسارع في مجال الذكاء الاصطناعي (AI)، تُعد قواعد البيانات كأداة أساسية لتخزين البيانات المنظمة التي تُستخدم في تدريب النماذج الذكية وتحليلها. معظم تطبيقات الذكاء الاصطناعي تعتمد بشكل مباشر على وجود بيانات دقيقة ومرتبة، مما يجعل تعلم قواعد البيانات خطوة أولى مهمة لمن يرغب في الدخول إلى عالم الذكاء الاصطناعي وتحليل البيانات.

تم إعداد هذا الكتاب ليكون دليلاً مبسطًا وشاملاً، يزود المتعلم بالأسس المعرفية اللازمة لتطبيق هذه المهارات في مشاريعه البرمجية، ويفتح له آفاقًا مُستقبلية لفهم تقنيات الذكاء الاصطناعي والتعامل معها بفعالية

المؤلفون

الوحـدة الأولى- المُعالجة الرقمية
قواعد البيانات SQLite في Python

1. مدخل إلى قواعد البيانات
   Introduction to Database
2. إنشاء قواعد البيانات
   Creating a Database
3. إنشاء الجداول
   Creating Tables
4. إضافة البيانات
   Inserting Data
5. الاستعلام عن البيانات
   Querying Data
6. تحديث البيانات
   Updating Data
7. حذف البيانات
   Deleting Data

قواعد البيانات SQLite
مدخل إلى قواعد البيانات
Introduction to Database

نواتج التعـلم

- التعرف على قواعد البيانات مع توضيح أهميتها، أنواعها، واستخداماتها.
- استعراض تطوّر نظم قواعد البيانات عبر المراحل التاريخية، وصولاً إلى تكاملها مع تقنيات الذكاء الاصطناعي الحديثة.
- تحليل مكونات قاعدة البيانات العلائقية، وشرح وظائفها بأمثلة متعددة.
- تطبيق المهـــارات العملية باستخـــدام برنامج DB Browser for SQLite لفتح قواعـــــد البيانات، معاينتها، وتحليل محتواها.
- تقييم الجوانب الأخلاقية المتعلقة باستخدام قواعد البيانات وضمـــــان حماية المعلومات.
- توظيف مهارات التحليل لاستخلاص معلومات دقيقة ومفـــيدة من قواعد بيانات فعلية.

يمثـل رمـز الاسـتجابة السـريعة QR رابـط
لملفـات أوراق العمـل، ومصـادر التعلـم

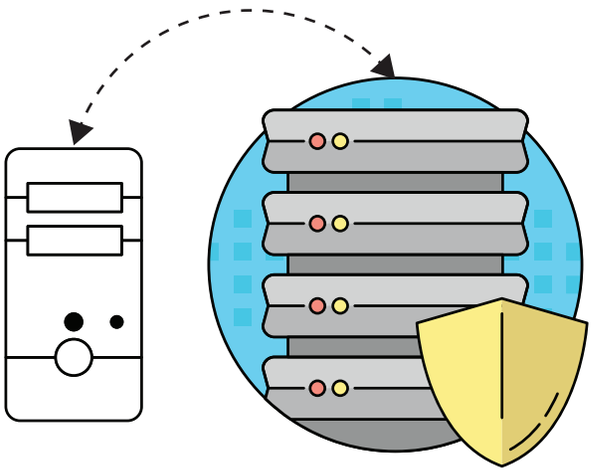
مقدمة في عالم البيانات

في العصر الرقمي الحالي، تُعد البيانات المحرك الأساسي للتقدم في شتى المجالات. إن القدرة على تخزين البيانات وإدارتها بكفاءة أصبحت ضرورة مُلحة. هنا تأتي قواعد البيانات لتؤدي دورًا محورياً في هيكلة المعلومات وتسهيل الوصول إليها والتعامل معها بسرعة ودقة تُعتبر قواعد البيانات أداة تُمكن من تنظيم الأفكار، إدارة المشاريع، وتحليل البيانات لاستخلاص رؤى جديدة، وهي بمثابة ذاكرة قوية، ذكية، ومُنظمة تمنح التطبيقات القدرة على معالجة كميات هائلة من المعلومات بكفاءة غير مسبوقة؛ حيث إن إتقان قواعد البيانات يمثل مفتاحاً لفهم جزء كبير من التكنولوجيا الحديثة وبوابة لإنشاء تطبيقات أكثر ذكاءً وقوة.

أهمية قواعد البيانات

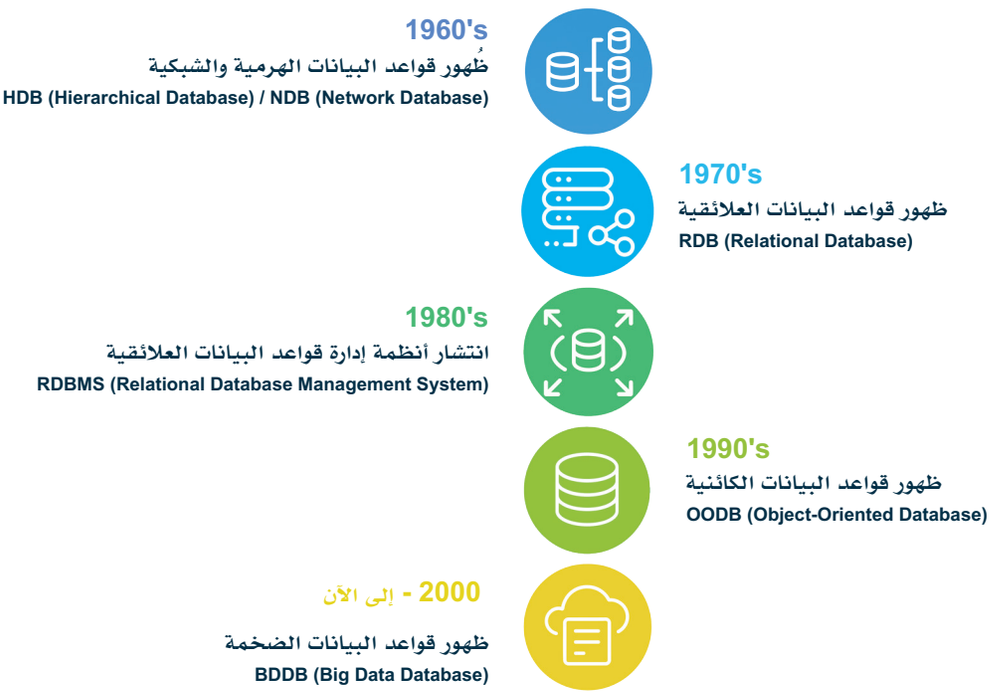
تُعد قواعد البيانات من الأدوات الأساسية في عالم التكنولوجيا الحديث، حيث تُستخدم في عدة مجالات لتنظيم البيانات، تخزينها وإدارتها بطريقة تتيح الوصول إليها وإدارتها بسهولة. وفيما يلي أهم الأسباب التي تجعل قواعد البيانات ضرورية:

- تنظيم البيانات.
- سهولة البحث، والاسترجاع.
- الحفاظ على دقة البيانات.
- إمكانية مشاركة البيانات.
- أمان المعلومات.
- إنشاء تقارير وتحليل النتائج.



نُبذه تاريخية عن قواعد البيانات

تطوّرت قواعد البيانات عبر عقود متلاحقة، بدأت في الستينيات مع استخدام أنظمة الملفات البسيطة، ثم ظهرت قواعد البيانات الهرمية والشبكية، تلاها في السبعينيات ظهور النموذج العلائقي الذي شكّل ثورة في إدارة البيانات. وفي الثمانينيات والتسعينيات، انتشرت أنظمة إدارة قواعد البيانات العلائقية (RDBMS) على نطاق واسع ثم ظهرت قواعد البيانات الكائنية (OODB) وقواعد بيانات NoSQL لدعم تطبيقات الويب، وصولاً إلى العصر الحديث حيث أصبحت قواعد البيانات الضخمة والمعتمدة على الذكاء الاصطناعي (AI-Powered Big Data DBs) جزءًا أساسيًا من التحول الرقمي والخدمات السحابية.



شكل رقم (1-1) يمثل التطور التاريخي لقواعد البيانات

مميزات قواعد البيانات

شكل رقم (1-2) يمثل أهم مميزات قواعد البيانات

استخدامات قواعد البيانات في الحياة اليومية

شكل رقم (1-3) أمثلة على استخدامات قواعد البيانات

الجوانب الأخلاقية الواجب مراعاتها عند التعامل مع قواعد البيانات

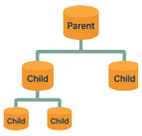
عند التعامل مع قواعد البيانات، من الضروري الالتزام بأخلاقيات تقنية المعلومات لحماية البيانات وضمان سلامتها. ومن أبرز الجوانب الأخلاقية ما يلي:

- احترام الخصوصية: الامتناع عن محاولة الوصول إلى بيانات شخصية أو حساسة دون تصريح قانوني أو مهني.
- النزاهة في التعامل مع البيانات: عدم تعديل أو حذف البيانات دون إذن، وتجنّب أي تلاعب يؤدي إلى تضليل أو إضرار.
- عدم إساءة مشاركة البيانات: الامتناع عن توزيع أو مشاركة ملفات قواعد البيانات خارج الأطر المصرّح بها.
- احترام حقوق الملكية: عدم نسخ أو استخدام قواعد البيانات أو التطبيقات المرتبطة بها دون إذن من المالك أو الجهة الرسمية.
- الحفاظ على سلامة النظام: عدم القيام بأي تصرف يؤدي إلى إتلاف البيانات أو الأنظمة المرتبطة بها.
- الامتثال للأنظمة والقوانين: الالتزام بالتشريعات المحلية والدولية الخاصة بحماية البيانات وحقوق المستخدمين.
- تحمل المسؤولية المهنية: التصرّف بمسؤولية وأمانة عند إدارة البيانات، وضمان توثيق أي تغيير أو تعديل يتم على النظام.

أنواع قواعد البيانات الأساسية

هنـاك عـدة أنـواع مـن قواعـد البيانـات، وكل نـوع يُسـتخدم حسـب طبيعـة البيانـات وطريقـة تنظيمهـا، مـن أبرزهـا:

- **قواعد البيانات الهرمية Hierarchical Databases**



تشـبه شـكل الشـجرة، حيـث ترتبـط البيانـات بطريقـة "الأب Parent - الابـن Child". كانـت تُسـتخدم فـي أنظمـة الحاسـوب القديمـة مثـل IMS مـن IBM.

- **قواعد البيانات الشبكية Network Databases**

هـي نـوع مـن قواعـد البيانـات الهيكليـة تُنظّـم فيهـا السـجلات Records علـى شـكل عُقـد Nodes مترابطـة عبـر مؤشـرات Pointers، ممـا يسـمح بتمثيـل العلاقـات متعـددة إلـى متعـددة Many-to-Many بشـكل مباشـر.

- **قواعد البيانات العلائقية Relational Databases**

تعتمـد علـى الجـداول والعلاقـات، وهـي الأكثـر اسـتخدامًا حاليًا، مثـل: Oracle، MySQL، SQLite، وهـي النـوع الـذي سـنتعرض لـه فـي هـذا الكتـاب

- **قواعد البيانات الكائنية Object-Oriented Databases**

تُسـتخدم لتخزيـن البيانـات في شـكل كائنـات Objects المسـتخدم فـي البرمجـة الكائنيـة التوجـه OOP، مثل لغـة Java أو C++.

- **قواعد بيانات NoSQL**

تتعامـل مـع كميـات ضخمـة مـن البيانـات غيـر المنظمـة، وتُسـتخدم فـي تطبيقـات الويـب والموبايـل مثـل MongoDB، وFirebase.

- **قواعد البيانات الموزعة Distributed Databases**

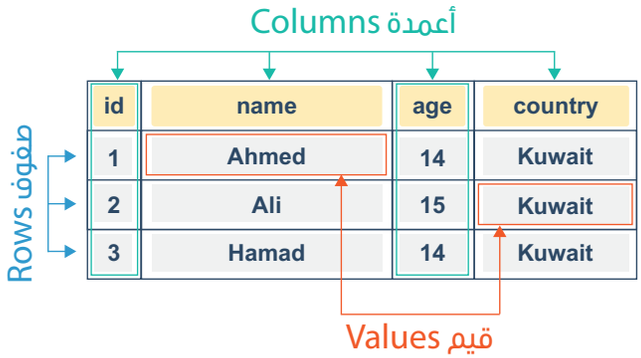
تكـون البيانـات مخزنـة علـى أكثـر مـن خـادم Server فـي أماكـن مختلفـة، لكنهـا تُـدار كأنهـا قاعـدة واحـدة.

مكونات قاعدة البيانات العلائقية

تُستخدم لتنظيم البيانات في جداول مترابطة، مما يسهل حفظها، البحث عنها، وتحديثها بشكل منظم وفعّال. تعتمد على العلاقات بين الجداول المختلفة، وتُعد من أكثر أنواع قواعد البيانات استخدامًا في العالم.

1. الجداول Tables:

تتكون قاعدة البيانات من جدول واحد أو أكثر؛ حيث يمثل الجدول كياناً معيناً (مثل جداول: بيانات الطلاب، الكتب، معالم إحدى الدول)، ويتكون الجدول من أعمدة وصفوف (سجلات بقاعدة البيانات).



شكل رقم (1-4) يوضح مُكونات جدول بقاعدة بيانات

2. الأعمدة Columns:

العمود هو وحدة بيانات رأسية في جدول قاعدة البيانات، يُمثل خاصية أو سمة معينة للكيانات التي يُمثلها الجدول، ويكون له اسم ونوع بيانات محدد.

3. الصفوف Rows:

الصف في قاعدة البيانات هو مكون أفقي في الجدول يُستخدم لتمثيل سجل واحد من البيانات، أي أنه يحتوي على مجموعة من القيم المرتبطة بكيان معين.

4. المفتاح الأساسي Primary Key:

قيد يُطبق على عمود أو مجموعة أعمدة ذات قيم فريدة في الجدول، لا يمكن أن تتكرر ولا أن تكون فارغة.

5. المفتاح الخارجي Foreign Key:

قيد يُطبق على عمود أو مجموعة أعمدة، يُستخدم لربط جدول بجدول آخر، يرتبط بمفتاح أساسي في جدول مختلف.

6. العلاقات Relationships:

الروابط التي تُنشأ بين الجداول المختلفة بهدف تنظيم البيانات وربطها بطريقة منطقية. فبدلاً من تكرار البيانات، نُقسّم المعلومات إلى جداول متخصصة، ونربطها معًا باستخدام المفاتيح مثل (المفتاح الأساسي والمفتاح الخارجي).

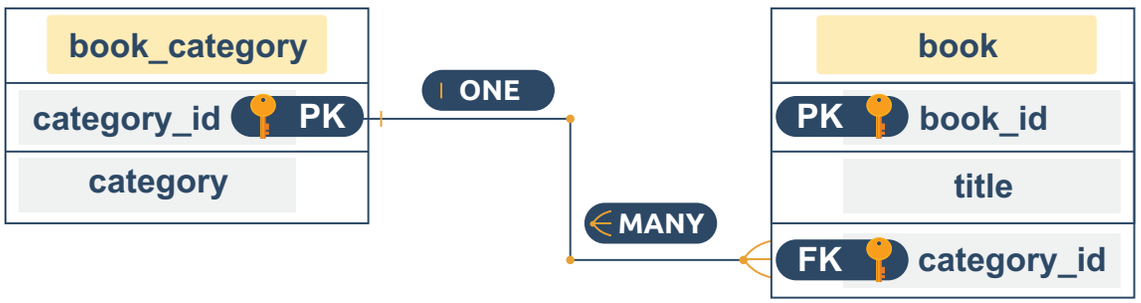
أنواع العلاقات بين الجداول

يوجد عدة أنواع للعلاقات تُستخدم لتنظيم الترابط بين الجداول بطريقة منطقية وهي:

- واحد إلى واحد One to One: يكون لكل صف في الجدول الأول (الرئيسي) صف واحد فقط مرتبط به في الجدول الثاني (الفرعي)، والعكس صحيح.

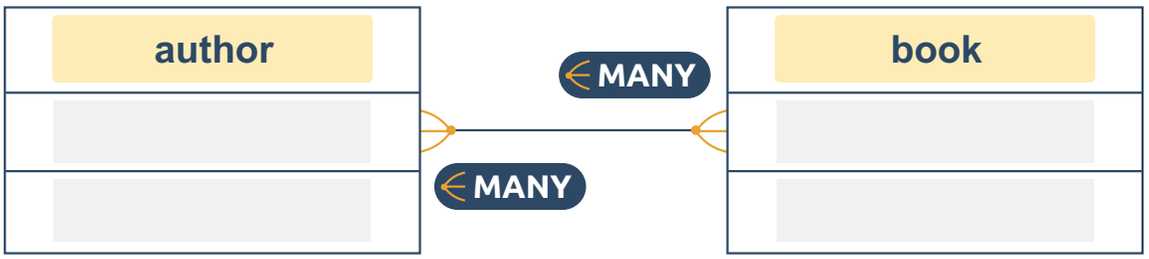
شكل رقم (1-5) يوضح العلاقة من نوع واحد إلى واحد

- واحد إلى مُتعدد One to Many: يمكن لكل صف في الجدول الأول (الرئيسي) أن يرتبط بعدة صفوف في الجدول الثاني (الفرعي)، بينما كل صف في الجدول الثاني يرتبط بصف واحد فقط في الجدول الأول.



شكل رقم (1-6) يوضح العلاقة من نوع واحد إلى مُتعدد

- متعدد إلى متعدد Many to Many: يُمكن لكل صف في الجدول الأول (الرئيسي) أن يرتبط بعدة صفوف في الجدول الثاني (الفرعي)، والعكس صحيح.



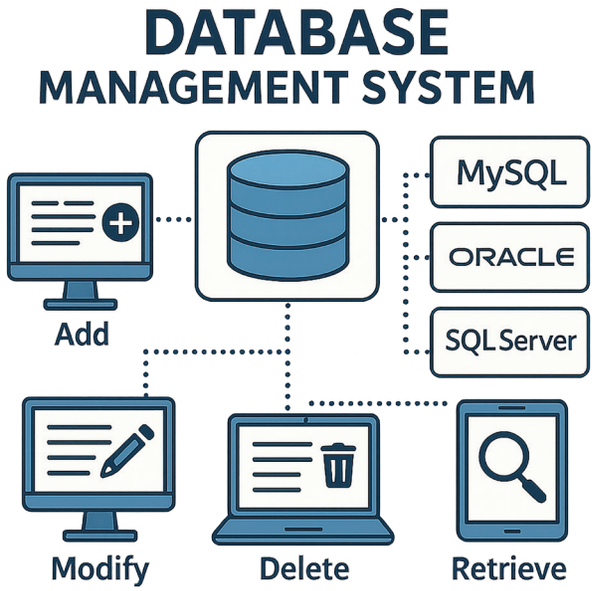
شكل رقم (1-7) يوضح العلاقة من نوع متعدد إلى متعدد

في بعض الحالات، يكون من الطبيعي أن يرتبط صف في جدولٍ واحدٍ بعدة صفوف في جدولٍ آخر، والعكس صحيح. على سبيل المثال، الكتاب يُمكن أن يكون له عدة مؤلفين، والمؤلف يُمكن أن يكون له عدة كتب. هذه العلاقة تُسمى Many to Many. لكن في هذه العلاقة لا يمكن الربط بين الجدولين بشكل مباشر، بل يتم إنشاء جدول وسيط بين الجدولين المرتبطين لتمثيل هذه العلاقة

شكل رقم (1-8) يوضح الجدول الوسيط بين جدولين مرتبطين بعلاقة متعدد إلى متعدد

نظام إدارة قواعد البيانات (DBMS)

يُقصد به إدارة وتشغيل قاعدة البيانات، مثل MySQL أو Oracle أو SQL Server. ويسمح بإضافة وتعديل وحذف واسترجاع البيانات بسهولة.



برنامج DB Browser for SQLite

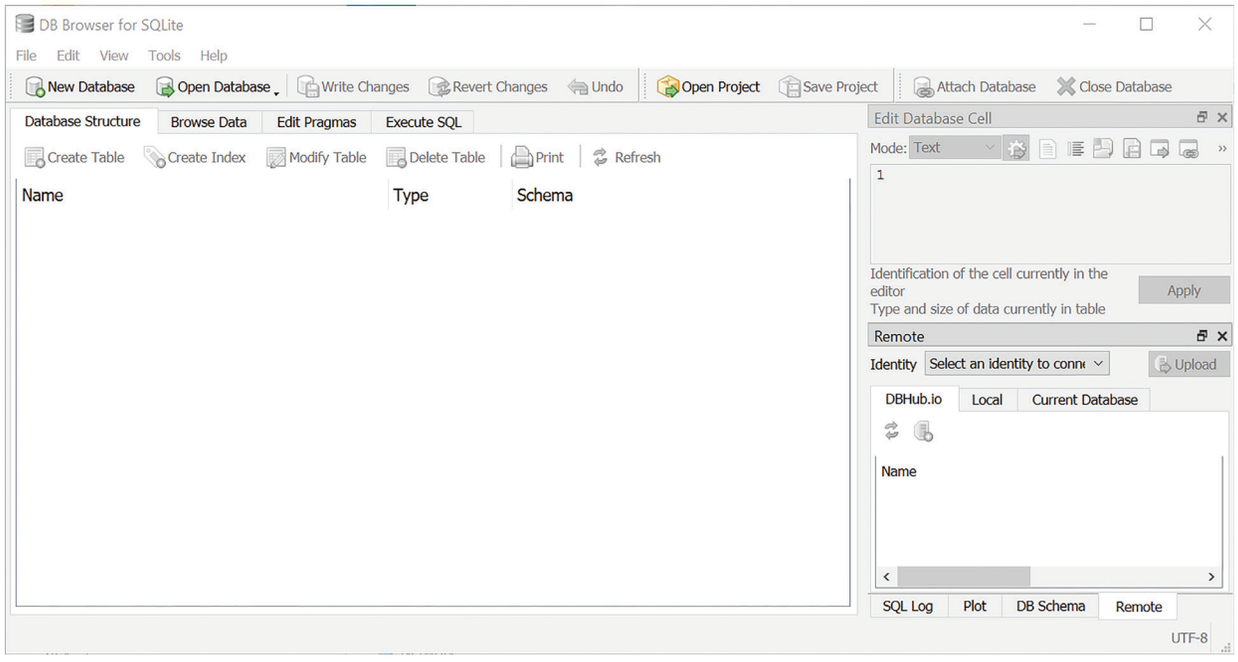
هو برنامج مجاني مفتوح المصدر، تُستخدم لإدارة قواعد البيانات من نوع SQLite. تتميز بواجهتها الرسومية البسيطة التي تساعد المستخدمين على إنشاء الجداول عبر إدخال وتعديل البيانات، وتنفيذ الاستعلامات المختلفة وعرض النتائج بطريقة مُنظمة تسهل قراءتها.

تحميل وتثبيت برنامج DB Browser for SQLite

- تحميل DB Browser for SQLite من الموقع الرسمي sqlitebrowser.org

- تثبيت البرنامج.

واجهة برنامج DB Browser for SQLite



شكل رقم (1-9) واجهة برنامج DB Browser for SQLite

شريط القوائم Menu Bar

يحتوي على العديد من القوائم الأساسية منها:

- File إدارة قواعد البيانات (فتح، حفظ، إنشاء، تصدير، ...).
- Edit التعامل مع الجداول (إنشاء، تعديل، حذف، ...)، بالإضافة إلى الإعدادات العامة.
- Tools أدوات متقدمة لإدارة قاعدة البيانات (التحقق من سلامة البيانات، ضغط البيانات، ...).

شريط الأدوات Toolbar

يوفر أدوات سريعة للعمليات الأساسية منها:

- New Database : إنشاء قاعدة بيانات جديدة.
- Open Database : فتح قاعدة بيانات سبق إنشاؤها.
- Write Changes : حفظ التعديلات.
- Close Database : إغلاق قاعدة البيانات.

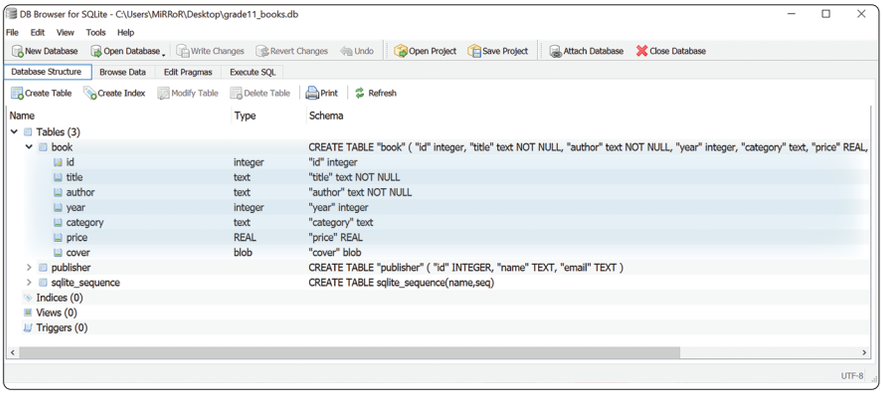
علامات التبويب الرئيسية Main Tabs

- Database Structure : عرض هيكلة قاعدة البيانات مثل الجداول والفهارس.
- Browse Data : استعراض بيانات الجداول، وإجراء التعديلات اللازمة مباشرةً.
- Edit Pragmas : تعديل إعدادات قاعدة البيانات.
- Execute SQL : كتابة وتنفيذ استعلامات SQL يدوياً.

التعامل مع برنامج DB Browser for SQLite

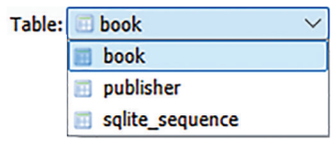
يُتيح برنامج DB Browser التعامل مع قواعد البيانات SQLite من خلال العديد من العمليات منها:

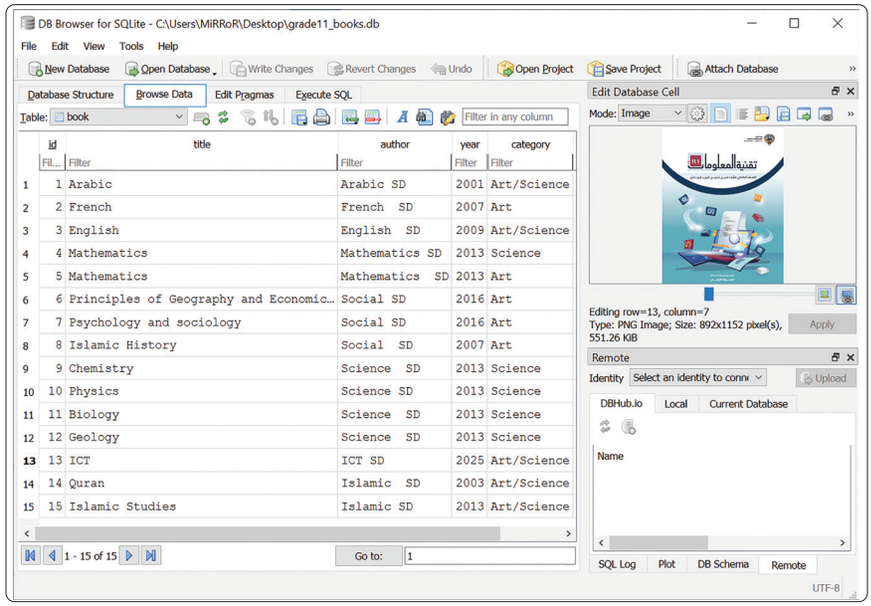
- **فتح قاعدة البيانات:** اختيار Open Database من قائمة File، أو بالضغط على Open Database من شريط الأدوات، ثم تحديد قاعدة البيانات من المجلد المخزنة به.
- **استعراض هيكلة الجدول:** اختر علامة التبويب Database Structure، ثم تحديد الجدول.



شكل رقم (1-10) يُوضح هيكلية الجدول book

- تعديل هيكلية الجدول: اختيار الأمر Modifiy Table من قائمة Edit.
- استعراض بيانات الجداول: اختر علامة التبويب Browse Data، ثم اختر من قائمة الجداول المنسدلة الجدول المناسب.





شكل رقم (1-11) يُوضح استعراض بيانات الجدول

- إغلاق قاعدة البيانات: اختر الأمر Close Database من قائمة File، أو بالضغط على Close Database من شريط الأدوات Toolbar.
- إغلاق البرنامج: اختر الأمر Exit من قائمة File.

ملاحظات:

- عند إجراء التعديلات على قاعدة البيانات يجب حفظ التغييرات بالضغط على Write Changes من شريط الأدوات Toolbar.
- بعد الانتهاء من العمل على قاعدة البيانات، يُفضل إغلاقها بطريقة صحيحة من خلال الضغط على Close Database من شريط الأدوات Toolbar. للتأكد من حفظ جميع التعديلات، وإغلاق الاتصال بقاعدة البيانات بأمان، ليقلل من احتمالية فقدان البيانات أو حدوث تلف في الملف.

أوراق العمل

ورقة عمل (1)

تضم دولة الكويت العديد من المعالم السياحية، التراثية، والدينية المُميزة، ونحتاج إلى حفظ معلوماتها في قاعدة بيانات رقمية، تتضمن أشهر تلك المعالم.

عاين بيانات الجدول landmark بقاعدة البيانات Kuwait_landmarks.db، وذلك باتباع الخطوات التالية:

1. شغل برنامج DB Browser for SQLite.
2. استدع قاعدة البيانات Kuwait_landmarks.db من مجلد أوراق العمل.
3. عاين هيكلية الجدول landmark:
   - أسماء الأعمدة. : .....

   .....
4. عاين بيانات الجدول landmark، وتعرف على مكوناته ثم استكمل ما يلي:
   - موقع "Al-Tahrir Tower" location : .....
   - سنة التأسيس "The Avenues mall" year : .....
   - عدد معالم دولة الكويت في الجدول : .....
   - فئة المَعلم "Souq Al-Mubarakiya" category : .....
5. أغلق قاعدة البيانات Kuwait_landmarks.db
6. إنهاء البرنامج.

قواعد البيانات SQLite في Python
إنشاء قواعد البيانات Creating a Database

نواتج التعـلم

- تعريف مفهوم SQLite ويبيّن أهميتها ومزاياها ضمن بيئة Python.
- شرح خطوات إنشاء قاعدة بيانات، والاتصال بها باستخدام دالة connect.
- تطبيق/ تنفيذ إنشاء قاعدة بيانات جديدة باستخدام لغة Python ضمن بيئة تطوير متكاملة (IDE) مثل PyCharm.
- كتابة تعليمات برمجية تشمل التعامل مع الاستثناءات عبر try/ except/ finally.
- الإدراك الكامل لأهمية إغلاق اتصال قاعدة البيانات من أجـــل ضمان تكامل البيانات وسلامتها ومنع فقدانها.

يُمثـل رمـز الاسـتجابة السـريعة QR رابـط لملفـات أوراق العمـل، ومصـادر التعلـم

إنشاء قواعد البيانات Creating a Database

تُستخدم قواعد البيانات لتخزين البيانات وتنظيمها ومعالجتها. وتُعد SQLite إحدى قواعد البيانات التي يُمكن إنشاؤها والتعامل معها بسهولة في لغة Python من خلال المكتبة القياسية sqlite3، دون الحاجة إلى خادم خارجي (Server).

صيغة التعليمة البرمجية للاتصال بقاعدة البيانات SQLite في Python

شكل رقم (2-1) يُوضح صيغة التعليمة البرمجية للاتصال بقاعدة البيانات

إنشاء قاعدة البيانات SQLite في Python

يتم إنشاء قاعدة بيانات جديدة باستخدام إحدى بيئات التطوير المتكاملة IDE (PyCharm) من خلال مجموعة من الخطوات الأساسية التالية:

1. كتابة التعليمات البرمجية.
   - استدعاء مكتبة sqlite3.
   - إنشاء اتصال بقاعدة البيانات.
   - إغلاق الاتصال بقاعدة البيانات.
2. تنفيذ التعليمات البرمجية.
3. تأكد من إنشاء ملف قاعدة البيانات بمجلد المشروع.

إنشاء قاعدة البيانات SQLite في Python

مثال 1: إنشاء قاعدة البيانات باسم grade11_books.db.

خطوات التنفيذ

- استدعاء مكتبة sqlite3.
- إنشاء / فتح اتصال بقاعدة البيانات.
- إغلاق الاتصال بقاعدة البيانات.

```
1  import sqlite3
2  connection= sqlite3.connect('grade11_books.db')
3  connection.close()
```

التفسير

import sqlite3

- استيراد مكتبة sqlite3.

---

connection= sqlite3.connect('grade11_books.db')

- إنشاء / فتح اتصال بقاعدة البيانات.
  - sqlite3.connect: تُستخدم دالة connect من مكتبة sqlite3 لإنشاء قاعدة البيانات باسم grade11_books.db.
  - connection: يُمثّل كائن (المتغير) الاتصال النشط بقاعدة البيانات، والذي يُتيح تنفيذ العمليات المختلفة عليها، مثل: إنشاء الجداول، إضافة البيانات، تعديلها، عرضها، وحذفها.

---

connection.close()

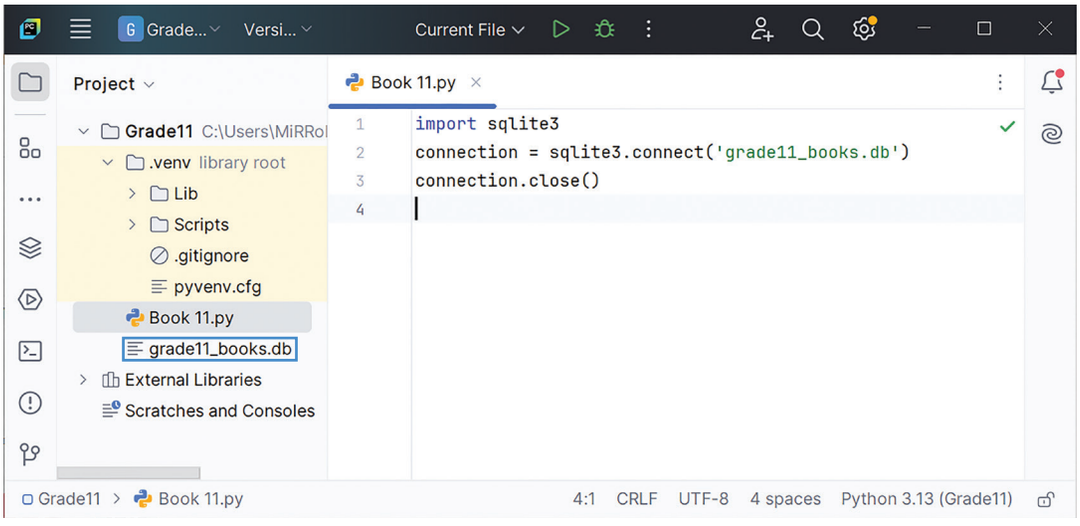
- إغلاق الاتصال بقاعدة البيانات.

ملاحظات:

- إذا كانت قاعدة البيانات غير موجودة يتم إنشاؤها تلقائياً.
- الامتداد الافتراضي لملف قاعدة البيانات من نوع SQLite هو db.
- يجب التأكد من إغلاق الاتصال بقاعدة البيانات بشكل صحيح، للأسباب التالية:
  - منع تسرب الموارد (مثل الذاكرة).
  - ضمان سلامة البيانات وعدم فقد البيانات.
  - تجنّب حدوث أخطاء قد تعيق فتح قاعدة البيانات مرة أخرى، نتيجة بقاء الاتصال بها مفتوحًا دون إغلاق.

Program Output

- في بيئة التطوير المتكاملة PyCharm IDE.
- يظهر ملف قاعدة البيانات grade11_books.db في Project Panel.



شكل رقم (2-2) يُوضح ملف قاعدة البيانات

إنشاء قاعدة البيانات SQLite في Python مع استخدام الاستثناءات

استخدام الاستثناءات try / except / finally لمعالجة الأخطاء أثناء التعامل مع قواعد البيانات.

مثال 2: إنشاء قاعدة البيانات باسم grade11_books.db، مع استخدام الاستثناءات.

خطوات التنفيذ

- استدعاء مكتبة sqlite3.
- تخصيص القيمة None للمتغير connection.
- إضافة تعليمة try لمعالجة الأخطاء.
  - إنشاء / فتح اتصال بقاعدة البيانات.
- إضافة تعليمة except.
  - تنفيذ أوامر محددة في حال حدوث خطأ.
- إضافة تعليمة finally.
  - يختبر وجود اتصال بقاعدة البيانات
    - تحقق الشرط: يتم إغلاق الاتصال بقاعدة البيانات.

```
1   import sqlite3
2   connection = None
3   try:
4       connection = sqlite3.connect('grade11_books.db')
5       print("Connected to the database successfully.")
6   except Exception as e:
7       print(f"An unexpected error occurred: {e}")
8   finally:
9       if connection is not None:
10          connection.close()
11      print("Database connection is closed.")
```

التفسير

```
import sqlite3
```

- استيراد مكتبة sqlite3.

```
connection = None
```

- إنشاء متغير باسم connection، وتخصيص القيمة None له، وهي قيمة غير محددة، أي أن المتغير لا يشير حالياً إلى أي اتصال حالي بقاعدة بيانات.

```
try:
    connection = sqlite3.connect('grade11_books.db')
    print("Connected to the database successfully.")
```

- إنشاء اتصال بقاعدة البيانات grade11_books.db، وإذا نجح الاتصال، يتم تعديل قيمة المتغير connection، ثم طباعة رسالة تؤكد ذلك.

```
except Exception as e:
    print(f 'An unexpected error occurred: {e}')
```

- إذا حدث خطأ أثناء محاولة الاتصال، يتم طباعة رسالة تُوضح الخطأ (المتغير e).

```
finally:
```

- في جميع الحالات (تم الاتصال بقاعدة البيانات بنجاح أم حدث خطأ أثناء التنفيذ).

```
    if  connection is not None:
```

- في حال وجود اتصال بقاعدة البيانات.

```
connection.close()
```

- يتم إغلاق قاعدة البيانات.

```
print("Database connection is closed.")
```

- يطبع رسالة إن الاتصال مغُلق.

ملاحظة:

تم تخصيص قيمة للمتغير connection في بداية التعليمات البرمجية، لضمان تعريف المتغير في جميع الحالات، سواء تم الاتصال بقاعدة البيانات بنجاح أم حدث خطأ أثناء التنفيذ

أوراق العمل

ورقة عمل (2)

أنشئ قاعدة بيانات Kuwait_landmarks.db لتوثيق معالم دولة الكويت، باتباع الخطوات التالية:

1. أنشئ مشروعًا جديدًا باسم landmark_database، مستخدمًا بيئة التطوير المناسبة IDE.
2. أنشئ ملف Python جديد باسم data_database.py.
   - استدع مكتبة sqlite3.
   - أنشئ اتصال بقاعدة البيانات.
   - أغلق الاتصال بقاعدة البيانات.
3. نفذ التعليمات البرمجية.
4. تأكد من إنشاء ملف قاعدة البيانات بمجلد المشروع.

- **تطوير البرنامج:** طور البرنامج ليشمل معالجة الاستثناءات.

قواعد البيانات SQLite في Python
إنشاء الجداول Creating Tables

نواتج التعـلم

- شرح مكونات الجدول في قواعد البيانات.
- التمييز بين أنواع البيانات المختلفة، واستخداماتها المناسبة.
- إنشاء جدولاً جديدًا وُفق هياكل محددة.
- تطبيق القيود المناسبة أثناء إنشاء الجداول.
- معاينة هيكلية الجدول من خلال برنامج DB Browser for SQLite.

يُمثـل رمـز الاسـتجابة السـريعة QR رابـط
لملفـات أوراق العمـل، ومصـادر التعلـم

إنشاء الجداول Creating Tables

تُعد الجداول هيكلاً أساسيًا في قواعد البيانات، تستخدم لتخزين، تنظيم البيانات، وتنفيذ العمليات الأساسية CRUD (الإدخال، الاستعلام، التعديل، الحذف) بسهولة وكفاءة.

يتكون الجدول من مجموعة من الصفوف Rows، والأعمدة Columns، بحيث يُمثّل كل صف سجل Record، ويُمثّل كل عمود خاصية Attribute.

Column Name
Primary Key
Row
Value

id	name	title	price
1	Mike Owens	The Definitive Guide to SQLite	41.39
2	Vivian Siahaan	Learn SQLite with Python	32.5
3	Allen Taylor	SQL All-in-One For Dummies	27.61
4	Michael Hernandez	Database Design for Mere Mortals	39.48
5	S. Basu	Learn SQLite with Python For Beginners	7.99

شكل رقم (3-1) يوضح المُكونات الأساسية لجدول بيانات الكتب الدراسية

أنواع البيانات Data Types في SQLite

تصنيفات تُحدد شكل وطبيعة المعلومات التي يمكن تخزينها في كل عمود من أعمدة الجدول. تساعد في تنظيم المعلومات والتأكد من التعامل معها بطريقة صحيحة عند التخزين أو المعالجة.

نوع البيانات	الوصف	مثال
INTEGER	تخزين الأرقام الصحيحة.	سنة النشر: 2025 - العمر: 17 - عدد الكتب: 1500 - عدد الطلاب: 20.
REAL	تخزين الأرقام العشرية.	سعر الكتاب: 3.15 - الوزن: 100.0 - المسافة: 0.75.
TEXT	تخزين النصوص.	اسم المؤلف: S. Ali. عنوان الكتاب: Learn SQLite.
BLOB	تخزين كائنات ثنائية كبيرة Binary Large Object.	الصور: JPEG, PNG, GIF. المستندات: PDF, Word, Excel. الملفات الصوتية / المرئية: MP3, MP4.
NULL	يستخدم لتمثيل القيم الفارغة (بدون قيمة).	عنوان الكتاب غير مُدخل - تاريخ طباعة الكتاب غير متوفر - البريد الإلكتروني فارغ

جدول رقم (3-1) يوضح التصنيفات الرئيسة لأنواع البيانات المستخدمة في SQLite.

عند إنشاء جدول، يجب أن يكون لكل عمود نوع بيانات أساسي مثل INTEGER أو TEXT أو REAL أو BLOB، أما NULL فهي ليست نوع بيانات للعمود، وإنما تُستخدم للدلالة على غياب القيمة.

القيود Constraints

قواعد تُطبق في الجدول لضمان صحة ودقة البيانات، وتُستخدم لمنع إدخال بيانات غير صحيحة أو غير مسموح بها، ويمكن إضافتها عند إنشاء الجدول أو تعديله

القيد Constraint	الوصف	الصيغة العامة column DATA_TYPE Constraint
NOT NULL	لا يسمح بترك العمود فارغًا.	name TEXT NOT NULL
UNIQUE	يضمن عدم تكرار القيم في العمود.	email TEXT UNIQUE
PRIMARY KEY	يحدد العمود كمفتاح أساسي، مما يجعله: • NOT NULL • UNIQUE	id INTEGER PRIMARY KEY
CHECK	يفرض شرطًا منطقيًا يجب أن يتحقق عند إدخال قيمة للعمود.	age INTEGER CHECK (age <= 20) name TEXT CHECK (length (name) < 30)

جدول رقم (3-2) يوضح القيود المستخدمة في الجدول في SQLite.

ملاحظات:

- يُمكن الجمع بين أكثر من قيد في نفس العمود.

```
name TEXT NOT NULL CHECK (length (name) < 30)
```

- AUTOINCREMENT عبارة عن سلوك خاص داخل SQLite، يُستخدم مع المفتاح الأساسي PRIMARY KEY من النوع INTEGER، ليتم زيادة القيمة تلقائيًا بمقدار واحد مع كل صف جديد.
- INTEGER PRIMARY KEY يزيد القيمة تلقائيًا بمقدار واحد، حتى بدون AUTOINCREMENT.
- يتم تحديد قيمة افتراضية باستخدام DEFAULT عند عدم إدخال قيمة للعمود.

```
nationality TEXT DEFAULT 'Kuwait'
```

ضوابط إنشاء الجداول، وتسمية الجداول والأعمدة في SQLite

ضوابط إنشاء الجدول

يجب عند إنشاء جدول تحديد التالي:

- **اسم الجدول:** اختيار اسمًا واضحًا ومعبرًا.
- **الأعمدة وأنواع البيانات:** تحديد أنواعها وخصائصها، وضبط القيود إن لزم ذلك.
- **المفتاح الأساسي:** تحديد عمود أو أعمدة المفتاح الأساسي.
- **العلاقات مع الجداول الأخرى:** إن وجدت.

القواعد القياسية لتسمية الجداول والأعمدة

- يحتوي اسم الجدول أو اسم العمود على أحرف، أرقام، الشرطة السفلية (_).
- يبدأ اسم الجدول أو اسم العمود بحرف (يُمكن أن يبدأ بشرطة سفلية _).
- لا يبدأ اسم الجدول أو اسم العمود برقم.
- لا يُسمح باستخدام المسافات أو الرموز الخاصة مثل @, #, $.
- اسم الجدول أو اسم العمود غير حساس لحالة الأحرف الإنجليزية (صغيرة أو كبيرة).
- يجب أن يكون لكل جدول اسم فريد لا يُمكن تكراره.
- يجب أن يكون لكل عمود بالجدول اسم فريد لا يُمكن تكراره.

صيغة التعليمة البرمجية لإنشاء جدول في قاعدة البيانات SQLite في Python

تنفيذ أمر SQL داخل قاعدة البيانات

أمر SQL لإنشاء جدول

عبارة شرطية اختيارية

اسم الجدول

```
cursor.execute (""" CREATE TABLE IF NOT EXISTS table_name (
                        column1  data_type constraints,
                        column2  data_type constraints,
                        . . .
                    )
""")
```

أسماء الأعمدة — نوع البيانات — قُيود اختيارية

شكل رقم (3-2) يوضح صيغة كتابة التعليمة البرمجية لإنشاء جدول

ملاحظة:

- المؤشر cursor هو كائن يُستخدم كأداة تساعد في التفاعل مع قاعدة البيانات بطريقة منظمة مثل (إنشاء قاعدة البيانات، وأيضًا إضافة، استرجاع، تعديل، وحذف البيانات).
- يُفضل استخدام العبارة الشرطية IF NOT EXISTS لمنع حدوث أخطاء عند محاولة إنشاء جدول موجود مسبقًا.

خطوات إنشاء جدول في قاعدة البيانات

1. تحديد بنية (هيكلية) الجدول:
   - اسم الجدول.
   - أسماء الأعمدة.
   - نوع البيانات لكل عمود.
   - القيود اللازمة لكل عمود (إن وجدت).
   - العلاقة مع الجداول الأخرى (إن وُجدت).
2. كتابة التعليمات البرمجية:
   - استدعاء مكتبة sqlite3.
   - إنشاء / فتح اتصال بقاعدة البيانات.
   - إنشاء كائن المُؤشر للتعامل مع قاعدة البيانات.
   - إنشاء جدول.
   - حفظ التغيرات.
   - إغلاق الاتصال.
3. تنفيذ التعليمات البرمجية.
4. مُعاينة بنية الجدول في برنامج DB Browser.

إنشاء جدول في قاعدة البيانات

مثال 1 : إنشاء جدول book في قاعدة البيانات grade11_books

البيانات التالية تُوضح الأعمدة، أنواعها، والقيود المطلوب تطبيقها على الجدول المراد إنشاؤه.

اسم العمود	النوع	القيود
id	INTEGER	المفتاح الأساسي PRIMARY KEY
title	TEXT	غير فارغ NOT NULL
author	TEXT	-
category	TEXT	-
year	INTEGER	-
price	REAL	-

جدول رقم (3-3) يوضح مثال عن إنشاء جدول في قاعدة البيانات.

خطوات التنفيذ

- استدعاء مكتبة sqlite3.
- إنشاء / فتح اتصال بقاعدة البيانات.
- إنشاء كائن المُؤشر للتعامل مع قاعدة البيانات.
- إنشاء جدول.
- حفظ التغيرات.
- إغلاق الاتصال.
- تنفيذ التعليمات البرمجية.
- مُعاينة بنية الجدول في برنامج Db Browser.

```
1   import sqlite3
2   connection = sqlite3.connect("grade11_books.db")
3   cursor = connection.cursor()
4   cursor.execute('''
5       CREATE TABLE IF NOT EXISTS book(
6           id          INTEGER    PRIMARY KEY,
7           title       TEXT       NOT NULL,
8           author      TEXT,
9           year        INTEGER,
10          category    TEXT,
11          price       REAL)
12  ''')
13  connection.commit()
14  connection.close()
```

التفسير

import sqlite3

- استيراد مكتبة sqlite3.

---

connection = sqlite3.connect ("grade11_books.db")

- الاتصال بقاعدة بيانات grade11_books.db.

  إذا كانت قاعدة البيانات غير موجودة، فسيتم إنشاؤها تلقائيًا.

---

cursor = connection.cursor()

- إنشاء كائن المؤشر cursor.

```
cursor.execute('''
        CREATE TABLE IF NOT EXISTS book(
                id          INTEGER  PRIMARY  KEY,
                title       TEXT       NOT  NULL,
                author      TEXT,
                year        INTEGER,
                category TEXT,
                price       REAL
         )
''')
```

إنشاء الجدول book.

```
connection.commit()
```

حفظ التغييرات.

commit() : دالة تُستخدم لحفظ التغييرات التي تم إجراؤها على قاعدة البيانات.

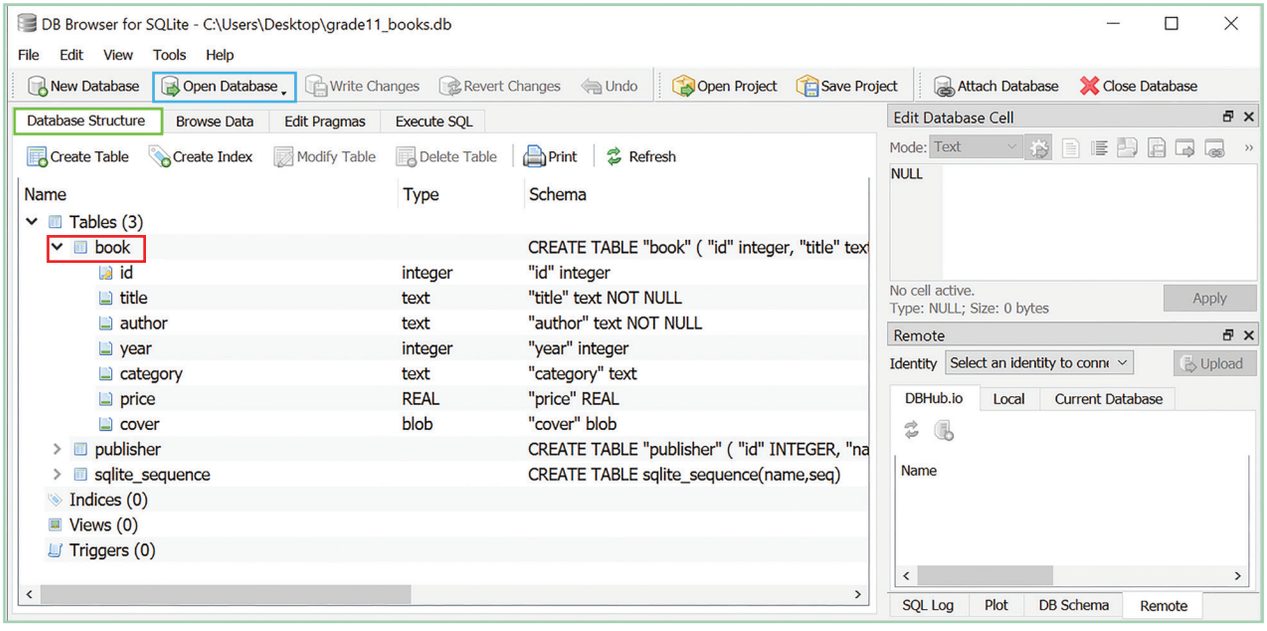
```
connection.close()
```

إغلاق الاتصال بقاعدة البيانات.

معاينة هيكلية الجدول Table Structure

يُتيح برنامج DB Browser for SQLite معاينة هيكلية الجدول، وذلك باتباع هذه الخطوات:

1. تشغيل برنامج DB Browser for SQLite.
2. اختيار الأداة Open Database.
3. تحديد قاعدة البيانات grade11_books.db من مجلد المشروع.
4. الانتقال إلى علامة التبويب Database Structure.
5. اختيار الجدول book من قائمة جداول قاعدة البيانات لتظهر تفاصيل هيكلية الجدول.



شكل رقم (3-3) يوضح خطوات معاينة هيكلية الجدول Table Structure

أوراق العمل

ورقة عمل (3)

إنشاء جدول جديد landmark بقاعدة البيانات Kuwait_landmarks.db **لتخزين بيانات بعض معالم دولة الكويت، مُتبعًا الخطوات التالية:**

1. استدع المشروع landmark_table.
2. افتح ملف data_table.py.

```
1  import sqlite3
2  connection = sqlite3.connect ('Kuwait_landmarks.db')
3  cursor = connection.cursor()
4
5
6
7
8
9
10
11
12
13 connction.commit()
14 connection.close()
```

3. استكمل التعليمة البرمجية لإنشاء جدول landmark، وفقًا للبيانات التالية:

اسم العمود	النوع	القيود
id	INTEGER	المفتاح الأساسي PRIMARY KEY
name	TEXT	غير فارغ NOT NULL
location	TEXT	-
category	TEXT	-
year	INTEGER	-
price	REAL	-

جدول (3-4) جدول landmark بقاعدة البيانات kuwait_landmarks.db.

4. نفذ التعليمات البرمجية.

5. عاين هيكلية الجدول باستخدام برنامج DB Browser.

قواعد البيانات SQLite في Python
إضافة البيانات Inserting Data

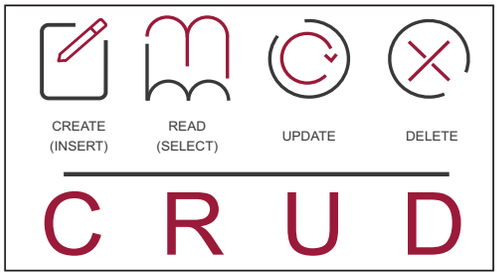
نواتج التعـلم

- التعرف على مفهوم عمليات CRUD، وكيفية تطبيقها على قواعد البيانات.
- إدراج سجل واحد في قاعدة البيانات باستخدام الدالة ()execute.
- إدراج عدة سجلات دفعة واحدة باستخدام الدالة ()executemany.
- تطوير تعليمات برمجية لاستقبال بيانات المستخدم وتخزينها في قاعدة البيانات.
- تطبيق آليات التعامل مع الاستثناءات لضمان سلامة البيانات وإغلاق الاتصـال بقاعدة البيانات بشكل صحيح.
- معاينة محتوى الجداول باستخدام برنامج DB Browser for SQLite.

يمثـل رمـز الاسـتجابة السـريعة QR رابـط
لملفـات أوراق العمـل، ومصـادر التعلـم

التعامل مع قواعد البيانات SQLite في Python

عند التعامل مع قواعد البيانات، هناك أربع عمليات (مهام) رئيسة تُستخدم بصورة متكررة (تُمثل بالاختصار CRUD):



- الإدخال INSERT: إضافة بيانات جديدة إلى قاعدة البيانات.
- القراءة SELECT: قراءة واسترجاع بيانات من قاعدة البيانات.
- التعديل UPDATE: تعديل أو تحديث بيانات مُخزنة في قاعدة البيانات.
- الحذف DELETE: حذف بيانات موجودة في قاعدة البيانات.

إضافة البيانات Inserting Data

تُعد عملية إضافة البيانات من العمليات الأساسية في إدارة قواعد البيانات حيث تتبع نهجًا منظمًا لتخزين البيانات في الجداول.

صيغة التعليمة البرمجية لإضافة البيانات SQLite في Python

```
cursor.execute ("INSERT INTO table_name (column1, column2, ...) VALUES (?, ?, ...)", (value1, value2, ...))
```

- يُنفذ أمر SQL داخل قاعدة البيانات
- اسم الجدول
- عناصر مؤقتة placeholders للقيم المطلوب إدخالها
- أمر SQL لإضافة بيانات للجدول
- أسماء الأعمدة
- القيم المطلوب إدخالها

شكل رقم (4-1) يوضح صيغة كتابة التعليمة البرمجية لإضافة البيانات

خطوات إدخال البيانات في قاعدة البيانات

1. تحديد القيم المطلوب إضافتها لإعمدة الجدول.
2. كتابة التعليمات البرمجية:
   - استدعاء مكتبة sqlite3.
   - إنشاء/ فتح اتصال بقاعدة البيانات.
   - إنشاء كائن المؤشر للتعامل مع قاعدة البيانات.
   - إضافة القيم للأعمدة.
   - حفظ التغييرات.
   - إغلاق الاتصال.
3. تنفيذ التعليمات البرمجية.
4. معاينة محتوى الجدول في برنامج DB Browser.

إدخال صف واحد إلى الجدول

مثال 1: إضافة البيانات التالية في جدول book بقاعدة البيانات grade11_books.db.

Column name	id	title	author	year	category	price
Data values		Arabic	Arabic SD	2001	Art/Science	NULL

جدول رقم (4-1) يوضح بيانات إدخال صف واحد إلى جدول book بقاعدة البيانات grade11_books.db.

خطوات التنفيذ

- استدعاء مكتبة sqlite3.
- إنشاء / فتح اتصال بقاعدة البيانات.
- إنشاء كائن المؤشر للتعامل مع قاعدة البيانات.
- إضافة البيانات.
- حفظ التغييرات.
- إغلاق الاتصال.
- تنفيذ التعليمات البرمجية.
- معاينة النتائج في برنامج DB Browser.

```
1   import sqlite3
2   # Connect to the database
3   connection= sqlite3.connect('grade11_books.db ')
4   cursor = connection.cursor()
5   # Insert data into table book
6   cursor.execute('''
7           INSERT INTO book(title, author,year,category,price)
8           VALUES( ?,?,?,?,?)''',
9           ('Arabic', 'Arabic SD', 2001, 'Art/Science',None)
10  )
11  # Commit changes
12  connection.commit()
13  # Close database
14  connection.close()
```

التفسير

```
import sqlite3
```

- استيراد مكتبة SQLite.

```
sqlite3.connect("grade11_books.db")
```

- الاتصال بقاعدة بيانات grade11_books.db.

```
cursor = connection.cursor()
```

- إنشاء كائن المؤشر cursor لتنفيذ أوامر SQL.

```
cursor.execute('''
        INSERT INTO book(title, author,year,category,price)
        VALUES( ?,?,?,?,?)''',
        ('Arabic','Arabic SD', 2001, 'Art/Science',None)
        )
```

- إضافة بيانات لأعمدة الجدول.
  - العنصر المؤقت placeholder (?) يتم استبداله بالقيم المطلوبة بطريقة آمنة.
  - المعاملات Parameters تُستخدم بدلًا من إدخال القيم مباشرة في الاستعلام، ذلك لحماية قاعدة البيانات ومنع هجمات الحقن البرمجي SQL Injection[1].
  - عند إرسال None إلى قاعدة بيانات SQLite، تحول تلقائيًا إلى NULL في قاعدة البيانات، وتُمثل الطريقة الصحيحة لإدخال القيم الفارغة في Python.

---

connection.commit()

- حفظ التغييرات في قاعدة البيانات.

---

connection.close()

- إغلاق الاتصال بقاعدة البيانات.

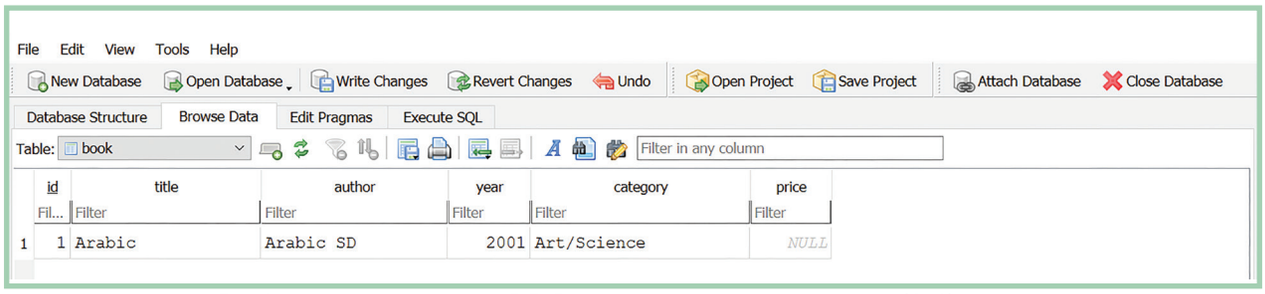
ملاحظات:

- تتولى قاعدة البيانات إسناد قيم رقمية تلقائية تصاعدية للمفتاح الأساسي من النوع INTEGER عند عدم إدخال قيم يدوية.
- لإضافة أكثر من صف للجدول، يتم تكرار التعليمة البرمجية cursor.execute("INSERT ....")
- عند تنفيذ الأمر INSERT دون تحديد اسم عمود أو أكثر، يتم تعبئة الأعمدة غير المحددة إما بقيمتها الافتراضية (إن وجدت) أو بالقيمة NULL، بشرط ألا يكون عليها قيد NOT NULL.
- يتم تمرير قيم الأعمدة في متغير من نوع tuple.
- عند إضافة قيمة واحدة للصف باستخدام tuple، يجب إضافة علامة الفاصلة , بعد القيمة، مثال ('AI_application' ,) أو (123 ,).

1 الحقن البرمجي SQL Injection هي نوع من الهجمات السيبرانية التي يستغل فيها المخترق مدخلات المستخدم غير الآمنة لإدخال أوامر SQL ضارة، بهدف الوصول إلى قاعدة البيانات أو تعديلها أو حذف بيانات منها دون إذن

استخدام برنامج DB Browser for SQLite

يُمكن استخدام برنامج DB Browser للتأكد من إضافة الصف إلى الجدول.



شكل رقم (4-2) يوضح استخدام برنامج DB Browser لمعاينة بيانات جدول Book.

إدخال سجلات متعددة إلى الجدول دفعة واحدة

تستخدم الدالة executmany() لإضافة مجموعة من الصفوف إلى الجدول.

```
cursor.executemany ('INSERT INTO table_name (column1, column2, ...) VALUES (? , ?, ...)',
                    [(value1_1, value1_2, ...), (value2_1, value2_2, ...), ... ]
                    )
```

مثال 2: إضافة البيانات التالية في جدول book بقاعدة البيانات grade11_books.db.

Columns name	id	title	author	year	category	price
Data values		ICT	ICT Supervision	2001	Art/Science	
		Physics	Science SD	2007	Science	1.500
		English	English SD	2009	Art/Science	

جدول رقم (4-2) يوضح بيانات إدخال صفوف متعددة إلى جدول book بقاعدة البيانات grade11_books.db.

```
1   import sqlite3
2   # Connect to the database
3   connection= sqlite3.connect("grade11_books.db")
4   cursor = connection.cursor()
5   # Insert data into table book
6   cursor.executemany('''
7           INSERT INTO book(title, author,year,category,price)
8           VALUES( ?,?,?,?,?)''',
9       [
10          ('ICT', 'ICT SD',2001,'Art/Science',None),
11          ('Physics', 'Science SD', 2007,'Science',1.500),
12          ('English', 'English SD',2009,'Art/Science',None)
13      ]
14  )
15  # Commit changes
16  connection.commit()
17  # Close database
18  connection.close()
```

التفسير

```
cursor.executemany ('''
        INSERT INTO book(title, author, year,category, price)
        VALUES (?,?,?,?,?)''',
         [
                ('ICT', 'ICT SD', 2001, 'Art/Science', None),
                ('Physics', 'Scince SD', 2007, 'Science', 1.500),
                ('English', 'English SD', 2009, 'Art/Science', None)
         ]
)
```

- استخدم الدالة cursor.executemany : لإدراج عدة سجلات (rows) في الجدول دفعة واحدة.
- استخدام قائمة (List) تتكون من عدة عناصر (كل عنصر من النوع tuple) :
  - ◄ عدد عناصر القائمة يساوي عدد الصفوف المطلوب إضافتها.

◄ كـل (tuple يمثل صفًا واحداً) يحتوي على القـيم المراد إضافتها إلى الأعمدة ,title , author year, category, price.

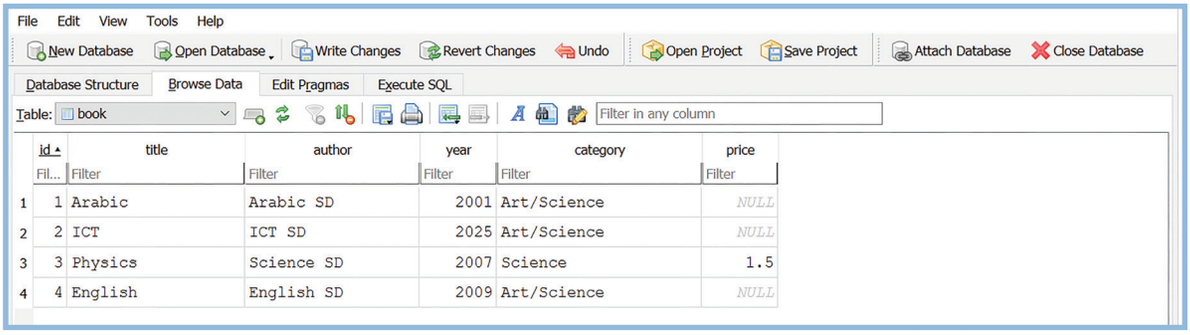
ملاحظات:

- يجـب اسـتخدام placeholders (؟) مـع الأمـر executemany لإدخـال مجموعـة مـن الصفـوف دفعـة واحـدة.
- يُمكن كتابة التعليمية البرمجية السابقة بالصورة التالية:

```
insert_sql ='INSERT INTO book(title, author, year,category, price) VALUES (?,?,?,?,?)'
book_list = [
      ('ICT', 'ICT SD', 2001, 'Art/Science', None),
      ('Physics', 'Science SD', 2007, 'Science', 1.500),
      ('English', 'English SD', 2009, 'Art/Science', None)
]
   cursor.executemany(insert_sql , book_list)
```

استخدام برنامج DB Browser for SQLite

يُمكن استخدام برنامج DB Browser للتأكد من إضافة الصفوف إلى الجدول.



	id	title	author	year	category	price
1	1	Arabic	Arabic SD	2001	Art/Science	NULL
2	2	ICT	ICT SD	2025	Art/Science	NULL
3	3	Physics	Science SD	2007	Science	1.5
4	4	English	English SD	2009	Art/Science	NULL

شكل رقم (3-4) يوضح استخدام برنامج DB Browser لمعاينة بيانات الصفوف جدول Book.

أوراق العمل

ورقة عمل (4)

إضافة سجل لجدول landmark بقاعدة البيانات Kuwait_landmarks.db، باتباع الخطوات التالية:

1. استدع المشروع landmark_insert.
2. افتح ملف data_insert1.py.

```
1   import sqlite3
2   # Connect to the database
3   connection = sqlite3.connect ('Kuwait_landmarks.db')
4   cursor = connection.cursor()
5   # Insert data into table book
6   cursor.execute(
7
8
9   )
10  # Commit changes
11  connection.commit()
12  # Close database
13  connection.close()
```

3. استكمل التعليمة البرمجية لإضافة سجل وفقًا للبيانات التالية:

Column names	id	name	location	category	year	price
Data values		Abdullah AlSalem Centre	Al-Shaab Area	Culture	2018	

جدول رقم (4-3) يوضح بيانات سجل مطلوب إضافته لجدول landmark بقاعدة البيانات Kuwait_landmarks.db.

4. نفذ التعليمات البرمجية.
5. عاين بيانات الجدول باستخدام برنامج DB Browser.

ورقة عمل (5)

إضافة عدة سجلات مرة واحدة لجـدول landmark بقاعدة البيانـــات Kuwait_landmarks.db، باتباع الخطوات التالية:

1. استدع المشروع landmark_insert.
2. افتح ملف data_insert2.py.

```
import sqlite3
connection = None
try:
    connection=sqlite3.connect('Kuwait_landmarks.db')
    cursor=connection.cursor()
    # Insert data into landmark table
    insert_sql="....."

    landmarks_list = [('Failaka Island', 'Off the eastern coast of Kuwait', 'Historical Landmark', 1950,5),
            ('Souq Al-Mubarakiya','Central Kuwait City','Traditional Market',1897,0),
            ('Museum of Modern Art','Salmiya Area','Art Museum',1980,1)]
    cursor.executemany(..... , .....)
    connection.commit()
except Exception as e :
    print("Error :", e)
finally :
    if connection is not None:
        connection.close()
```

3. استكمل التعليمة البرمجية لإضافة الصفوف المخصصة للمتغير landmarks_list.
4. نفذ التعليمات البرمجية.
5. عاين بيانات الجدول باستخدام برنامج DB Browser.

قواعد البيانات SQLite في Python
الاستعلام عن البيانات Querying Data

نواتج التعلم

- تعريف مفهوم الاستعلام (Database Query) لاسترجاع البيانات من قواعد البيانات.
- توضيح مفهوم الشروط المنطقية (Logical Expressions) وأهميتها في تحديد الصفوف المطلوب التعامل معها.
- التمييز بين الشروط البسيطة (Simple Conditions)، والمركبة (Compound Conditions).
- التعرف على العوامل الشرطية المختلفة في SQLite وكيفية استخدامها.
- كتابة استعلامات SQL في بيئة Python باستخدام مكتبة sqlite3.
- تنفيذ الاستعلامات واسترجاع النتائج باستخدام دوال fetch المختلفة.
- تطبيق خطوات إنشاء وتنفيذ استعلام في SQLite ضمن بيئة تطوير Python.
- عرض نتائج الاستعلام باستخدام الحلقات التكرارية.

يمثل رمز الاستجابة السريعة QR رابط لملفات أوراق العمل، ومصادر التعلم

الاستعلام عن البيانات Querying Data

يُعد الاستعلام أحد العمليات الرئيسية في إدارة قواعد البيانات، ويُستخدم لتصفية (اختيار / استرجاع) البيانات وُفق شروط ومعايير محددة.

الشروط في SQLite

هي التعبيرات المنطقية Logical Expressions التي تُستخدم لتحديد الصفوف التي يجب التعامل معها عند تنفيذ أوامر مثل (الاسترجاع، الحذف، التحديث) على قاعدة البيانات.

أنواع الشروط المستخدمة في الاستعلام

- الشرط البسيط Simple Condition

هو تعبير منطقي يتكون من مقارنة واحدة بين قيمتين، مثل (price> 30)، باستخدام عامل مقارنة واحد مثل (=، <، >، =<، =>، <>).

- الشرط المُركب Compound Condition

يتكون من شرطين أو أكثر يتم الربط بينهما باستخدام عوامل المنطق (AND, OR).

العوامل الشرطية في SQLite

تُستخدم العوامل الشرطية في SQLite لتحديد الشروط التي يجب أن تتحقق عند استرجاع البيانات، أو تحديثها، أو حذفها داخل قاعدة البيانات. وتُعد هذه العوامل أداة مهمة للتحكم في نتائج الاستعلامات، ومن أبرزها:

النوع: المساواة

العامل الشرطي: =

مثال:

price = 20

النوع: الاختلاف

العوامل الشرطية: != <>

مثال:

price != 20 price <> 20

النوع: أكبر من / أصغر من

العوامل الشرطية: < >

مثال:

price < 20 price > 20

النوع: أكبر من أو يساوي/ أصغر من أو يساوي

العوامل الشرطية: <= >=

مثال:

price <= 20 price >= 20

النوع: داخل النطاقات

العوامل الشرطية:

BETWEEN value1 AND value2

مثال:

price BETWEEN 10 AND 20

النوع: خارج النطاقات

العوامل الشرطية:

NOT BETWEEN value1 AND value2

مثال:

price NOT BETWEEN 10 AND 20

النوع: ضمن قائمة قيم

العوامل الشرطية:

IN (value1, value2, value3, ...)

مثال:

price IN (10,20,25)

النوع: خارج قائمة قيم

العوامل الشرطية:

NOT IN (value1, value2, value3, ...)

مثال:

price NOT IN (10,20,25)

النوع: التحقق من القيم الفارغة

العوامل الشرطية:

IS NULL

مثال:

title IS NULL

النوع: التحقق من القيم غير الفارغة

العوامل الشرطية:

IS NOT NULL

مثال: title IS NOT NULL

title IS NOT NULL AND price IS NOT NULL

النوع: يُطابق نمط جزئي نصي

العوامل الشرطية:

LIKE 'pattern' ⟶

% : تُمثل مجموعة أحرف مجهولة _ : تُمثل حرف مجهول

مثال: 'com%' LIKE title عنوان (كتاب) يبدأ بـ com

النوع: لا يُطابق نمط جزئي نصي

العوامل الشرطية:

NOT LIKE 'pattern' ⟶

مثال: 'com%' NOT LIKE title عنوان (كتاب) لا يبدأ بـ com

شكل رقم (5-1) يوضح العوامل الشرطية في SQLite

ملاحظات:

- توجد فروقات في صياغة العبارات الشرطية بين لغة Python وقواعد بيانات SQLite فكلٌ منهما يستخدم طريقة خاصة به لكتابة الشروط.
- القيمة NULL تشير إلى "عدم وجود قيمة" وهي ليست صفر (0) أو سلسلة نصية فارغة ("").

صيغة التعليمة البرمجية لإنشاء استعلامات SQLite في Python

شكل رقم (5-2) يوضح صيغة كتابة التعليمة البرمجية لإنشاء استعلامات

خطوات إنشاء استعلام في قاعدة البيانات

1. تحديد البيانات المطلوب استرجاعها.

- تحديد العمود / الأعمدة المطلوب استرجاعها.
  - إذا لم يتم تحديد الأعمدة وتم استخدام الرمز *، يتم استرجاع جميع قيم أعمدة الجدول.
- تحديد الشرط (المعيار) لتنفيذ الاستعلام بناءً عليه.
  - إذا لم يتم تحديد الشرط (بدون تعبير WHERE)، فسيتم استرجاع جميع صفوف الجدول.

2. كتابة التعليمات البرمجية.

- استدعاء مكتبة sqlite3.
- إنشاء / فتح اتصال بقاعدة البيانات.
- إنشاء كائن المؤشر للتعامل مع قاعدة البيانات.
- استرجاع البيانات.
- طباعة النتائج (الصفوف).
- إغلاق الاتصال.

3. تنفيذ التعليمات البرمجية.

4. معاينة النتائج (طباعة الصفوف في PyCharm).

الاستعلام عن جميع بيانات الجدول

مثال 1: تحديد جميع بيانات الجدول book بقاعدة البيانات grade11_books.

خطوات التنفيذ

- استدعاء مكتبة sqlite3.
- إنشاء / فتح اتصال بقاعدة البيانات.
- إنشاء كائن المؤشر للتعامل مع قاعدة البيانات.
- استرجاع البيانات.
- طباعة النتائج (الصفوف).
- إغلاق الاتصال.
- تنفيذ التعليمات البرمجية.
- معاينة النتائج (طباعة الصفوف في PyCharm).

```
1   import sqlite3
2   # Connect to the database
3   connection = sqlite3.connect('grade11_books.db')
4   cursor = connection.cursor()
5   # Data Retrieval
6   cursor.execute("SELECT * FROM book")   #(*) means all fields
7   books=cursor.fetchall()                # Fetches all records(books) As list of tuples
8   # print records in separate lines
9   for book in books:
10      print(book)
11  #close database
12  connection.close()
```

التفسير

```
cursor.execute("SELECT * FROM book")
```

- * SELECT تسترجع جميع الأعمدة من جدول book.
- عدم وجود عبارة (WHERE): يعني استرجاع جميع الصفوف المُخزنة في الجدول دون تصفية.

```
books = cursor.fetchall()
```

- تسترجع جميع النتائج الناتجة عن الاستعلام وتخزنها في متغير books في الذاكرة.
- نوع المُتغير books: قائمة (List) تحتوي على صفوف (Tuples)، يُمثل كل صف سجلًا من الجـدول.

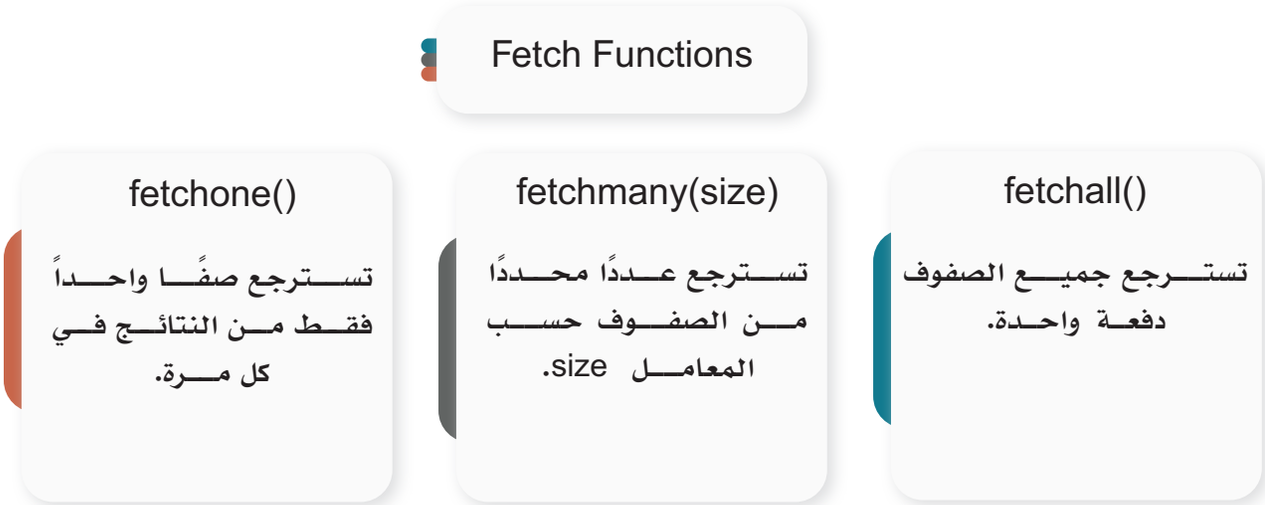
```
for book in books:
    print(book)
```

- تقوم الحلقة التكرارية بطباعة كل صف من النتائج في سطر مستقل.

ناتج تنفيذ التعليمة البرمجية (البيانات التي تم استرجاعها)

```
(1, 'Arabic', 'Arabic SD', 2001, 'Art/Science', 1.5)
(2, 'French', 'French SD', 2007, 'Art', 1.5)
(3, 'English', 'English  SD', 2009, 'Art/Science', 1.5)
(4, 'Mathematics', 'Mathematics SD', 2013, 'Science', 1.5)
(5, 'Mathematics', 'Mathematics SD', 2013, 'Art', 1.5)
(6, 'Principles of Geography and Economics\n', 'Social SD', 2016, 'Art', 1.5)
(7, 'Psychology and sociology', 'Social SD', 2016, 'Art', 1.5)
(8, 'Islamic History', 'Social SD', 2007, 'Art', 1.5)
(9, 'Chemistry', 'Science SD', 2013, 'Science', 1.5)
(10, 'Physics', 'Science SD', 2013, 'Science', None)
(11, 'Biology', 'Science SD', 2013, 'Science', 1.5)
(12, 'Geology', 'Science SD', 2013, 'Science', 1.5)
(13, 'ICT', 'ICT SD', 2025, 'Art/Science', 1.5)
(14, 'Quran', 'Islamic SD', 2003, 'Art/Science', 1.5)
(15, 'Islamic Studies', 'Islamic SD', 2013, 'Art/Science', 1.5)
```

طرق استخدام دوال fetch



شكل رقم (5-3) يوضح طرق استخدام دوال fetch

- الدالة fetchone تُعيد النتائج على شكل tuple.
- الدالتين fetchmany وfetchall تعيدان النتائج على شكل list يحتوي على عناصر من نوع tuple (كل tuple يُمثل صفًا من قاعدة البيانات).
- الدالة fetchall قد تستهلك الكثير من الذاكرة إذا كانت النتائج كبيرة.
- عند التعامل مع عدد كبير من الصفوف، يُنصح باستخدام fetchmany أو fetchone داخل حلقة لتقليل استهلاك الذاكرة وتحسين الأداء.
- دوال fetch تجلب البيانات فعليًا إلى Python بعد تنفيذ الاستعلام باستخدام الأمر SELECT.
- عند استخدام التعليمات التالية يتم حفظ جميع الصفوف المُسترجعة في متغير variable_name.

```
variable_name = cursor.fechall()
variable_name = cursor.fetchmany(size)
variable_name = cursor.fetchone()
```

الاستعلام عن عمود واحد أو أكثر لجميع الصفوف

مثال 2: تحديد بيانات عمودي العنوان title، والسعر price لجميع صفوف الجدول book بقاعدة البيانات grade11_books.db.

خطوات التنفيذ

- استدعاء مكتبة sqlite3.
- إنشاء / فتح اتصال بقاعدة البيانات.
- إنشاء كائن المؤشر للتعامل مع قاعدة البيانات.
- استرجاع البيانات.
- طباعة النتائج (الصفوف).
- إغلاق الاتصال.
- تنفيذ التعليمات البرمجية.
- معاينة النتائج (طباعة الصفوف في PyCharm).

```
1   import sqlite3
2   # Connect to the database
3   connection = sqlite3.connect('grade11_books.db')
4   cursor = connection.cursor()
5   # Data Retrieval
6   cursor.execute("SELECT title, price FROM book")
7   books=cursor.fetchall()
8   for book in books:
9       print(book)
10  #close database
11  connection.close()
```

التفسير

```
cursor.execute("SELECT title, price FROM book")
```

- تنفيذ استعلام SQL لاسترجاع عمودين فقط من جدول book، وهما:
  - title: عنوان الكتاب.
  - price: سعر الكتاب.

```
books = cursor.fetchall()
```

- استرجاع جميع النتائج الناتجة عن الاستعلام.
- يتم تخزين النتائج في متغير books على شكل قائمة (list) عناصرها tuples، كل tuple يمثل صفًا (أي كتابًا) يحتوي على قيمتي العنوان والسعر فقط.

```
for book in books:
    print(book)
```

- تقوم الحلقة التكرارية بطباعة كل صف من النتائج في سطر مستقل.

Program Output

```
('ICT', 1.5)
('Arabic', 1.5)
('English', None)
('French', 1.5)
('Chemistry', None)
('Islamic Studies', 1.5)
```

الاستعلام عن بيانات بناءً على شرط محدد

مثال 3: تحديد بيانات الصفوف (الكتب) وفق الشرط (طُبعت بعد عام 2010)، من جدول book بقاعدة البيانات grade11_books.db.

خطوات التنفيذ

- استدعاء مكتبة sqlite3.
- إنشاء / فتح اتصال بقاعدة البيانات.
- إنشاء كائن المؤشر للتعامل مع قاعدة البيانات.
- استرجاع البيانات.
- طباعة النتائج (الصفوف).
- إغلاق الاتصال.
- تنفيذ التعليمات البرمجية.
- معاينة النتائج (طباعة الصفوف في PyCharm).

```
1   import sqlite3
2   # Connect to the database
3   connection = sqlite3.connect('grade11_books.db')
4   cursor = connection.cursor()
5   # Data Retrieval
6   cursor.execute("SELECT * FROM book WHERE year>?",(2010,))
7   books=cursor.fetchall()
8   for book in books:
9       print(book)
10  #Close database
11  connection.close()
```

التفسير

```
cursor.execute("SELECT * FROM book WHERE year > ?",(2010,))
```

- تُستخدم SELECT * لاسترجاع جميع الأعمدة وفقًا لشرط الاستعلام.
- تُستخدم عبارة WHERE year > 2010 لتحديد شرط الاستعلام بحيث يتم استرجاع الصفوف الخاصة بالكتب التي تم طباعتها بعد عام 2010 فقط. (تم استبدال ? بالقيمة **2010**).

```
books = cursor.fetchall()
```

- تُسترجع جميع الصفوف المطابقة للشرط السابق وتُخزن في المتغير books.
- النتيجة تكون في قائمة (list) تحتوي على tuples، كل tuple يُمثل صف كتاب (جميع أعمدته).

```
for book in books:
    print(book)
```

- تقوم الحلقة التكرارية بطباعة كل صف من النتائج في سطر مستقل.

Program Output

```
(1, 'ICT', 'ICT SD', 2025, 'Art/Science', 1.5)
(5, 'Chemistry', 'Science SDn', 2013, 'Science', None)
(6, 'Islamic Studies', 'Islamic SD', 2013, 'Art/Science', 1.5)
```

أوراق العمل

ورقة عمل (6)

الاستعلام عن جميع البيانات من الجدول landmark بقاعدة البيانات Kuwait_landmarks.db مستخدمًا التعليمة fetchall، مُتبعًا الخطوات التالية:

1. استدعِ المشروع landmark_select.
2. افتح ملف data_select1.py.

```
1   import sqlite3
2   # Connect to the database
3   connection = sqlite3.connect('Kuwait_landmarks.db')
4   cursor = connection.cursor()
5   # Data Retrieval
6   cursor.execute(                                    )
7   rows = cursor.fetchall()
8   for row in rows:
9
10  # Close the database
11  connection.close()
```

3. استكمل التعليمات البرمجية لتنفيذ التالي:
   - استعلم عن جميع صفوف الجدول.
   - اطبع الصفوف التي تم استرجاعها (الاستعلام عنها).
4. نفذ التعليمات البرمجية.
5. عاين نتائج الاستعلام.
   - عدد الصفوف: .....

ورقة عمل (7)

الاستعلام عن بيانات محددة من الجدول landmark بقاعدة البيانات Kuwait_landmarks.db مُستخدمًا التعليمة fetchall، مُتبعًا الخطوات التالية:

1. استدعِ المشروع landmark_select.
2. افتح ملف data_select2.py.

```
1   import sqlite3
2   # Connect to the database
3   connection = sqlite3.connect('Kuwait_landmarks.db')
4   cursor = connection.cursor()
5   # Data Retrieval
6   cursor.execute(                                    )
7   # Fetch the Data
8
9   # Output the Data
10
11
12  # Close the database
13  connection.close()
```

3. استكمل التعليمات البرمجية اللازمة لتنفيذ الاستعلامات التالية:
   - استعلم عن عمودي (اسم مَعلم الكويت name، وسنة الإنشاء year) لجميع صفوف المعالم الثقافية Cultural فقط.
   - استخدم دالة fetchall() لاسترجاع البيانات وتخزينها في متغير records.
   - اكتب التعليمة البرمجية للحلقة التكرارية للمتغير records لطباعة كل صف في سطر منفصل.
4. نفذ التعليمات البرمجية.
5. عاين نتائج الاستعلام.

قواعد البيانات SQLite في Python
تحديث البيانات Updating Data

نواتج التعـلم

- شرح مفهوم تعديل البيانات في جداول قواعد البيانات باستخدام الأمر UPDATE.
- التمييز بين تحديث عمود أو أكثر في سجل واحد أو أكثر.
- كتابة تعليمات برمجية لتحديث البيانات باستخدام مكتبة sqlite3 في Python.
- تطبيق شروط بسيطة ومركبة لتحديث بيانات محددة.
- استخدام الدالة cursor.rowcount لمعرفة عدد الصفوف التي تم تعديلها.

يمثـل رمـز الاسـتجابة السـريعة QR رابـط
لملفـات أوراق العمـل، ومصـادر التعلـم

تحديث البيانات Updating Data

تُعدّ عملية تعديل البيانات من المهام الأساسية التي تهدف إلى تحديث أو تصحيح المعلومات المخزّنة ضمن جداول قاعدة البيانات. تشمل هذه العملية تعديل قيمة واحدة أو أكثر في صف معين، أو في مجموعة من الصفوف.

ولتنفيذ عملية التعديل في بيئة SQLite نستخدم الأمر UPDATE.

صيغة التعليمة البرمجية لتحديث قاعدة البيانات SQLite في Python

شكل رقم (6-1) يوضح صيغة كتابة التعليمة البرمجية لتحديث البيانات

خطوات تحديث بيانات قاعدة البيانات

1. تحديد البيانات المطلوب تحديثها :

- تحديد العمود / الأعمدة المطلوب تحديثها.
- تحديد القيمة / القيم الجديدة.
- تحديد الشرط للصفوف المطلوب تحديثها.

ملاحظة:

- يتم تحديث جميع الصفوف في الجدول إذا لم يتم تحديد شرط.

2. كتابة التعليمات البرمجية:
   - استدعاء مكتبة sqlite3.
   - إنشاء / فتح اتصال بقاعدة البيانات.
   - إنشاء كائن المؤشر للتعامل مع قاعدة البيانات.
   - تحديث البيانات.
   - حفظ التغييرات.
   - إغلاق الاتصال.
3. تنفيذ التعليمات البرمجية.
4. معاينة محتوى الجدول في برنامج DB Browser.

تحديث بيانات عمود واحد بناءً على شرط بسيط.

مثال 1: تعديل عمود السعر price في جدول book بقاعدة البيانات grade11_books.db.
لتصبح قيمته 2 دينار وفقًا للشرط id = 5.

خطوات التنفيذ

- استدعاء مكتبة sqlite3.
- إنشاء / فتح اتصال بقاعدة البيانات.
- إنشاء كائن المؤشر للتعامل مع قاعدة البيانات.
- تحديث البيانات.
- حفظ التغييرات.
- إغلاق الاتصال.
- تنفيذ التعليمات البرمجية.
- معاينة النتائج في برنامج DB Browser.

```
1   import sqlite3
2   connection = sqlite3.connect('grade11_books.db')
3   cursor = connection.cursor()
4   # Update data
5   cursor.execute('''
6           UPDATE book
7           SET price =?
8           WHERE id = ? ''',(2,5)
9   )
10  connection.commit()
11  connection.close()
```

التفسير

import sqlite3

- استيراد مكتبة SQLite للتعامل مع قواعد البيانات.

---

sqlite3.connect("grade11_books.db")

- الاتصال بقاعدة بيانات grade11_books.db.

---

cursor = connection.cursor()

- إنشاء كائن المؤشر cursor لتنفيذ أوامر SQL.

---

cursor.execute('UPDATE book SET price = ? WHERE id = ?',(2,5))

- تحديث قيمة العمود price لتصبح 2 في الصف الذي رقمه id = 5 داخل جدول book.

connection.commit()

- حفظ التغييرات في قاعدة البيانات.

---

connection.close()

- إغلاق الاتصال بقاعدة البيانات.

تحديث بيانات أكثر من عمود بالجدول بناء على شرط (معيار) بسيط.

مثال 2: تعديل بيانات أعمدة جدول book بقاعدة البيانات grade11_ books.db.
ليصبح price = 2.5 و year = 2024 وفقًا للشرط category = "Art".

خطوات التنفيذ

- استدعاء مكتبة sqlite3.
- إنشاء / فتح اتصال بقاعدة البيانات.
- إنشاء كائن المؤشر للتعامل مع قاعدة البيانات.
- تحديث البيانات.
- طباعة عدد الصفوف التي تم تحديثها.
- حفظ التغييرات.
- إغلاق الاتصال.
- تنفيذ التعليمات البرمجية.
- معاينة النتائج في برنامج DB Browser.

```
1   import sqlite3
2   connection = sqlite3.connect('grade11_books.db')
3   cursor = connection.cursor()
4   # Update data
5   cursor.execute('''
6           UPDATE book
7           SET price =?, year = ?
8           WHERE category = ? ''',(2.5,2024,'Art')
9   )
10  print("Number of rows updated = ", cursor.rowcount)
11  connection.commit()
12  connection.close()
```

التفسير

```
import sqlite3
```

استيراد مكتبة SQLite للتعامل مع قواعد البيانات.

```
sqlite3.connect("grade11_books.db")
```

الاتصال بقاعدة بيانات grade11_books.db.

```
cursor = connection.cursor()
```

إنشاء كائن المؤشر cursor لتنفيذ أوامر SQL.

```
cursor.execute('UPDATE book SET price =?, year = ? WHERE category = ?',(2.5,2024,'Art'))
```

تحديث قيم العمود price لتصبح 2.5، وقيم العمود year لتصبح 2024 لجميع الصفوف وفقًا للشرط category = "Art".

```
print(”Number of rows updated = ”, cursor.rowcount)
```

cursor.rowcount معرفة عدد الصفوف التي تم تحديثها.

```
connection.commit()
```

حفظ التغييرات في قاعدة البيانات.

```
connection.close()
```

إغلاق الاتصال بقاعدة البيانات.

ملاحظة:

- استخدم الخاصية cursor.rowcount لمعرفة عدد الصفوف التي تأثرت بآخر عملية (إضافة / تعديل/ حذف) تمت على قاعدة البيانات.

في المثال السابق:

تُستخدم التعليمة البرمجية التالية لتحديث جميع الصفوف التي ليس لها قيمة محددة بعمود تاريخ النشر.

```
cursor.execute("""
    UPDATE book
    SET price=?, year = ?
    WHERE year IS NULL""",(2.5,2024)
)
```

تحديث بيانات عمود بناء على شرط مركب.

التعليمة البرمجية التالية لتعديل عمود السعر price ليصبح 3 لجميع الكتب وفقًا للشرط المُركب (price < 2.5 (و) category ='Art').

```
cursor.execute('''
    UPDATE book
    SET price = ?
    WHERE  price < ? AND category = ?''',(3,2.5,'Art')
)
```

تحديث بيانات أكثر من عمود بناءً على شرط مركب.

التعليمة البرمجية التالية تُستخدم لتعديل قيمة عمود التصنيف category لتصبح Science، وقيمة عمود السعر price لتصبح 3 لجميع الكتب وفقًا للشرط المُركب:

(price = 2.5 و category = 'Art').

```
cursor.execute('''
    UPDATE book
    SET category = ?, price = ?
    WHERE category = ? AND price < ?''',('Science', 3, 'Art', 2.5)
)
```

أوراق العمل

ورقة عمل (8)

تعديل / تحديث بيانات الجدول landmark بقاعدة البيانات Kuwait_landmarks.db، باتباع الخطوات التالية:

1. استدعِ المشروع landmark_update.

2. افتح ملف data_update1.py.

```
1   import sqlite3
2   connection = sqlite3.connect ('Kuwait_landmarks.db')
3   cursor = connection.cursor()
4   # Update Data
5   cursor.execute(
6
7
8
9   )
10  print("Number of rows updated:", cursor.rowcount)
11  connection.commit()
12  connection.close()
```

3. استكمل التعليمات البرمجية لتنفيذ التالي:

- حدث بيانات الصف (id=1) لتصبح قيمة تذكرة الدخول تساوي دينارين (price = 2).

4. نفذ التعليمات البرمجية.

- عدد الصفوف التي تم تعديلها = ...

ورقة عمل (9)

تعديل / تحديث بيانات الجدول landmark بقاعدة البيانات Kuwait_landmarks.db، باتباع الخطوات التالية:

1. استدعِ المشروع landmark_update.
2. افتح ملف data_update2.py.

```
1  import sqlite3
2  connection = sqlite3.connect ('Kuwait_landmarks.db')
3  cursor = connection.cursor()
4  # Update Data
5  cursor.execute(
6
7
8
9  )
10 print("Number of rows updated:", cursor.rowcount)
11 connection.commit()
12 connection.close()
```

3. استكمل التعليمات البرمجية لتنفيذ التالي:
   - تحديث جميع صفوف المعالم الثقافية أو التاريخية
   - category = 'Cultural', category = 'Historical' ليصبح سعر الدخول مجاني (price=0).
4. نفذ التعليمات البرمجية.
   - عدد الصفوف التي تم تعديلها = ...

قواعد البيانات SQLite في Python
حذف البيانات Deleting Data

نواتج التعـلم

- شرح مفهوم عملية حذف البيانات DELETE، وأهميتها في قواعد البيانات.
- تنفيذ عملية حذف صف أو عدة صفوف محددة من جدول في قاعدة البيانات باستخدام أمر DELETE مع شرط WHERE.
- تطبيق حذف جميع الصفوف من جدول باستخدام أمر DELETE بدون شرط عند الحاجة.
- استخدام أسلوب البرمجة الآمنة من خلال المعاملات placeholders لتفـادي أخطاء تنفيذ الحذف وهجمات SQL Injection.
- عرض نتائج عملية الحذف باستخدام برنامج DB Browser for SQLite.

يمثـل رمـز الاسـتجابة السـريعة QR رابـط
لملفـات أوراق العمـل، ومصـادر التعلـم

حذف البيانات Deleting Data

تُعتبر عملية حذف البيانات من العمليات الاساسية في إدارة قواعد البيانات، حيث تهدف إلى إزالة سجل محدد أو مجموعة من الصفوف من الجداول بشكل دائم.

ولتنفيذ هذه العملية في SQLite يتم استخدام أمر DELETE مع تحديد الشروط اللازمة لضمان دقة الحذف.

صيغة التعليمة البرمجية لحذف البيانات SQLite في Python

شكل رقم (7-1) يوضح صيغة كتابة التعليمة البرمجية لحذف البيانات

خطوات حذف بيانات قاعدة البيانات

1. تحديد البيانات المطلوب حذفها:

- تحديد الشرط للصفوف المطلوب حذفها.
- تحديد عمود / أعمدة الشرط.
- تحديد القيمة / القيم.

ملاحظة:

تُحذف جميع الصفوف في الجدول، إذا لم يتم تحديد شرط الحذف.

2. كتابة التعليمات البرمجية:
   - استدعاء مكتبة sqlite3.
   - إنشاء / فتح اتصال بقاعدة البيانات.
   - إنشاء كائن المؤشر للتعامل مع قاعدة البيانات.
   - حذف البيانات.
   - طباعة عدد الصفوف التي تم حذفها.
   - حفظ التغييرات.
   - إغلاق الاتصال.

3. تنفيذ التعليمات البرمجية.

حذف الصفوف بناء على شرط بسيط.

مثال 1: حـذف الصفوف من جدول book بقاعدة البيانات grade11_books.db وفقـاً للشرط (title = 'Mathematics').

خطوات التنفيذ

- استدعاء مكتبة sqlite3.
- إنشاء / فتح اتصال بقاعدة البيانات.
- إنشاء كائن المؤشر للتعامل مع قاعدة البيانات.
- حذف البيانات.
- طباعة عدد الصفوف التي تم حذفها.
- حفظ التغييرات.
- إغلاق الاتصال.
- تنفيذ التعليمات البرمجية.

```
1   import sqlite3
2   connection = sqlite3.connect('grade11_books.db')
3   cursor = connection.cursor()
4   # Delete data
5   cursor.execute("""
6       DELETE FROM book
7       WHERE title=?""",
8       ('Mathematics',)
9   )
10  print("Number of rows deleted = ", cursor.rowcount)
11  connection.commit()
12  connection.close()
```

التفسير

```
import sqlite3
```

- استيراد مكتبة SQLite للتعامل مع قواعد البيانات.

```
connection = sqlite3.connect('grade11_books.db')
```

- الاتصال بقاعدة بيانات grade11_books.db.

```
cursor = connection.cursor()
```

- إنشاء كائن المؤشر cursor لتنفيذ أوامر SQL داخل قاعدة البيانات.

```
cursor.execute("DELETE FROM book WHERE title=?",('Mathematics',))
```

- حـذف الصفوف مـن جـدول book عند تحقق الشرط 'title = 'Mathematics.

```
print("Number of rows deleted = ", cursor.rowcount)
```

- cursor.rowcount معرفة عدد الصفوف التي تم حذفها.

```
connection.commit()
```

- حفظ التغييرات التي أُجريت على قاعدة البيانات.

```
connection.close()
```

- إغلاق الاتصال بقاعدة البيانات.

حذف الصفوف بناء على شرط مُركب.

مثال 2: حذف الصفوف من جدول book بقاعدة البيانات grade11_books.db، وفقًا للشرط (price > 10 ، و year < 2000).

خطوات التنفيذ

- استدعاء مكتبة sqlite3.
- إنشاء / فتح اتصال بقاعدة البيانات.
- إنشاء كائن المؤشر للتعامل مع قاعدة البيانات.
- حذف البيانات.
- طباعة عدد الصفوف التي تم حذفها.
- حفظ التغييرات.
- إغلاق الاتصال.
- تنفيذ التعليمات البرمجية.
- معاينة النتائج في برنامج DB Browser.

```
1  import sqlite3
2  connection = sqlite3.connect('grade11_books.db')
3  cursor = connection.cursor()
4  # Delete data
5  cursor.execute('DELETE FROM book WHERE price > ? AND year < ?',(10,2000))
6  print("Number of rows deleted = ", cursor.rowcount)
7  connection.commit()
8  connection.close()
```

التفسير

```
import sqlite3
```

- استيراد مكتبة SQLite للتعامل مع قواعد البيانات.

```
connection = sqlite3.connect('book.db')
```

- إنشاء اتصال بقاعدة البيانات book.db.

```
cursor = connection.cursor()
```

- إنشاء كائن المؤشر cursor لتنفيذ أوامر SQL داخل قاعدة البيانات.

```
cursor.execute("DELETE FROM book WHERE price > ? AND year < ?",(10,2000))
```

- حذف الصفوف من جدول «book» إذا كان السعر أكبر من 10 وسنة النشر أقل من 2000.

```
print("Number of rows deleted = ", cursor.rowcount)
```

- cursor.rowcount معرفة عدد الصفوف التي تم حذفها.

```
connection.commit()
```

- حفظ التغييرات في قاعدة البيانات.

```
connection.close()
```

- إغلاق الاتصال بقاعدة البيانات.

ملاحظة:

- حذف الصفوف التي لا يحتوي عمود تاريخ النشر (year) فيها على قيم.

```
cursor.execute('DELETE FROM book WHERE year IS NULL')
```

- لحذف جميع الصفوف غير الفارغة نستخدم المعامل الشرطي (IS NOT NULL).

أوراق العمل

ورقة عمل (10)

حـــذف صفـــوف مـــن الجـــدول landmark بقاعـــدة البيانـــات Kuwait_landmarks.db، باتبـــاع الخطـــوات التاليـــة:

1. استدعِ المشروع landmark_delete.
2. افتح ملف data_delete1.py.

```
1  import sqlite3
2  # Connect to the database
3  connection = sqlite3.connect ('Kuwait_landmarks.db')
4  cursor = connection.cursor()
5  #Delete Data
6  cursor.execute(
7
8
9
10 )
11 print("Number of rows deleted:", cursor.rowcount)
12 # Commit changes
13 connection.commit()
14 # Close database
15 connection.close()
```

3. استكمل التعليمات البرمجية:
   - احذف جميع صفوف معالم الكويت الثقافية. (category = "Cultural").
4. نفذ التعليمات البرمجية.
   - عدد الصفوف التي تم حذفها = ...

ورقة عمل (11)

حـــذف صفـــوف مـــن الجـــدول landmark بقاعـــدة البيانـــات Kuwait_landmarks.db، باتبـــاع الخطـــوات التاليـــة:

1. استدعِ المشروع landmark_delete.

2. افتح ملف data_delete2.py.

```
1  import sqlite3
2  # Connect to the database
3  connection = sqlite3.connect ('Kuwait_landmarks.db')
4  cursor = connection.cursor()
5  #Delete Data
6  cursor.execute(
7
8
9
10 )
11 print("Number of rows deleted:", cursor.rowcount)
12 # Commit changes
13 connection.commit()
14 # Close database
15 connection.close()
```

3. استكمل التعليمات البرمجية:

- احذف الصفوف التي تاريخ إنشائها year قبل العام 2000م، وسعر تذكرة الدخول price أكبر من دينار واحد.

4. نفذ التعليمات البرمجية.

- عدد الصفوف التي تم حذفها = ...

الوحـــدة الثانية
المُنتجات الرقمية Digital Products

1 الذكاء الاصطناعي وتكامله مع قواعد البيانات
مشروع تقنيات كشف الوجوه

2 مراحل تصميم وتطوير المنتج الرقمي

المنتجات الرقمية

الذكاء الاصطناعي وتكامله مع قواعد البيانات

AI and Database Integration

نواتج التعـلم

- شرح المفاهيم الأساسية لتقنيات الذكاء الاصطناعي ورؤية الحاسوب.
- الربط بين خوارزميات التعرف على الوجوه وقواعد البيانات في مشروع تطبيقي.
- إنشاء برنامجًا يستخدم مكتبات Python لتسجيل وتحليل البيانات البيومترية.

يُمثـل رمـز الاسـتجابة السـريعة **QR** رابـط
لملفـات أوراق العمـل، ومصـادر التعلـم

مدخل إلى مشروع تقنية التعرف على الوجوه

- **التعرف على الوجوه Face Recognition**

أحد تطبيقات رؤية الحاسوب Computer Vision التي تتيح تحليل الصور لاكتشاف وتمييز وجوه الأشخاص، سواء للتسجيل أو الأمان أو التحليل.

- **قواعد البيانات Databases**

وسيلة مُنظمة لتخزين المعلومات واسترجاعها. نستخدم في هذا المشروع SQLite لأنها خفيفة ولا تحتاج إلى خادم Server خارجي.

- **توظيف تقنيات الذكاء الاصطناعي مع قواعد البيانات Integrating AI with Databases**

استخدام Python لربط خوارزميات التعرف على الوجوه مع قاعدة بيانات SQLite لتخزين معلومات الأشخاص الذين تم التعرف عليهم مثل: الاسم، التاريخ، الوقت، وصورة الوجه.

المكتبات المستخدمة في المشروع

المكتبة	الاستخدام	تثبيت المكتبة
OpenCV	لاكتشاف الوجوه والتعامل مع الصور	pip install opencv-python
os	لإدارة الملفات ومسارات الصور	مكتبة مدمجة في Python
sqlite3	لإنشاء وتخزين البيانات في قاعدة بيانات محلية	
datetime	لتسجيل الوقت والتاريخ لكل وجه يتم التعرف عليه	

جدول رقم (8-1) يُوضح أسماء المكتبات المُستخدمة في المشروع

مكتبة OpenCV

Open-Source Computer Vision Library

مكتبة مفتوحة المصدر تحتوي على العديد من الخوارزميات والأدوات البرمجية لمعالجة الصور والرؤية الحاسوبية، وتدعمها مؤسسة OpenCV.

مراحل المشروع

إعداد بيئة المشروع

- تجهيز مكتبات Python مثل cv2 و sqlite3.
- إعداد مجلد يحتوي على صور الأشخاص المعروفين.
- التعرف على الوجه.
- استخدام مكتبة OpenCV لتحويل الصور إلى تدرجات رمادية.
- استخدام خوارزميات مطابقة النماذج مثل cv2.matchTemplate لمقارنة صورة الوجه المُدخل مع الصور المحفوظة في مجلد المشروع.

استخراج معلومات الوجه

- عند وجود تطابق بنسبة عالية (مثلاً > 0.7)، يتم اعتبار الوجه معروفًا.
- استخراج اسم الشخص من اسم ملف الصورة (مثلاً Ali.png يصبح الاسم "Ali").

الاتصال بقاعدة البيانات

- فتح اتصال بقاعدة بيانات SQLite.
- إنشاء جدول (إن لم يكن موجودًا) لتخزين بيانات الأشخاص المُتعرف عليهم.

تخزين البيانات

- إدخال بيانات مثل (الاسم، الوقت) إلى الجدول باستخدام جملة INSERT INTO.

بناء قاعدة البيانات

```
CREATE TABLE IF NOT EXISTS attendance (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    name TEXT NOT NULL,
    date TEXT,
    time TEXT
)
```

حيث إنشاء جدول attendance يتضمن الأعمدة التالية:

- رقم تعريفي تلقائي.
- اسم الشخص.
- تاريخ التعرف.
- وقت التعرف.

بناء البرنامج

1 تعريف دالة التعرف على الوجوه.

```
def recognize_face_by_template(face_gray):
```

- تقوم هذه الدالة بمقارنة الوجه المقتطع من الصورة بالصور المحفوظة في مجلد known_faces.
- تستخدم خوارزمية matchTemplate من OpenCV التي تقيس درجة التشابه بين صورتين.

منطق المقارنة:

```
result = cv2.matchTemplate(known_image, face_resized, cv2.TM_CCOEFF_NORMED)
```

- score يمثل نسبة التشابه.
- إذا تجاوزت النسبة 0.3 أو 0.7، تعتبر مطابقة مقبولة.

2 إنشاء قاعدة بيانات SQLite.

```
connection = sqlite3.connect("face_detections.db")
cursor = connection.cursor()
```

◄ يتم إنشاء قاعدة بيانات باسم face_detections.db.

```
CREATE TABLE IF NOT EXISTS face_log (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    image_name TEXT,
    person_name TEXT,
    x INTEGER,
    y INTEGER,
    w INTEGER,
    h INTEGER,
    detected_at TEXT
)
```

◄ الجدول face_log يحفظ اسم الشخص، موقع الوجه داخل الصورة، وتاريخ الاكتشاف.

3 قراءة الصورة من المستخدم.

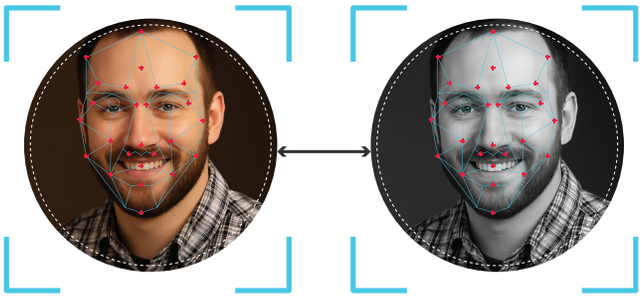
```
image_path = input("Enter image path: ").strip()
image = cv2.imread(image_path)
```

- يحصل البرنامج على مسار الصورة من المستخدم.
- يتحقق من إمكانية تحميل الصورة.

4 تحويل الصورة إلى تدرج رمادي.

```
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
```

- التعرف على الوجه يتم بشكل أكثر كفاءة باستخدام صور رمادية بدلًا من الألوان.



5 كشف الوجوه باستخدام كاشف Haar.

```
face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades +
'haarcascade_frontalface_default.xml')
faces = face_cascade.detectMultiScale(gray, scaleFactor=1.1,
minNeighbors=5)
```

- يستخدم كاشف جاهز من مكتبة OpenCV.
- يعيد مجموعة إحداثيات (x, y, w, h) لكل وجه مكتشف.

6 التعامل مع كل وجه مكتشف.

لكل وجه:

- يُقص الوجه من الصورة.
- يُستخدم في محاولة التعرف على وجه محدد recognize_face_by_template.
- تُرسم مستطيلات وتوضع أسماء على الصورة.
- تُخزّن البيانات في SQLite.

```
cv2.rectangle(image, (x, y), (x + w, y + h), (2 ,(0 ,255 ,0)
cv2.putText(image, name, (x, y + h + 25), ...)
```

7 حفظ النتائج.

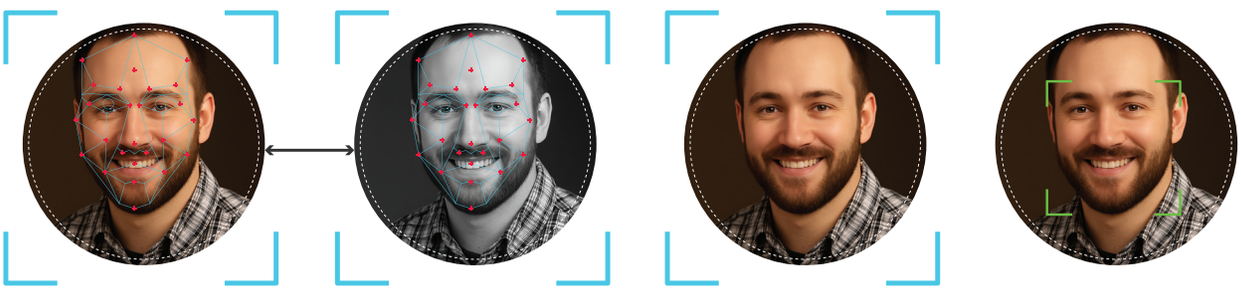
```
cursor.execute("""
   INSERT INTO face_log (...)
   VALUES (?, ?, ?, ?, ?, ?, ?)
""",(...))
connection.commit()
connection.close()
```

◄ يتم تسجيل كل عملية تعرف على الوجه مع الوقت والتاريخ في قاعدة البيانات.

8 عرض الصورة المُعدلة.

```
cv2.imshow("Identified Faces", image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

◄ تعرض الصورة بعد تمييز الوجوه والتعرف عليها.



التعليمات البرمجية المقترحة لتنفيذ المشروع

```
import cv2
import os
import sqlite3
from datetime import datetime

# Function to recognize a face using template matching
def recognize_face_by_template(face_gray):
  best_match_name = "Unknown"
  highest_score = 0

  for filename in os.listdir("known_faces"):
    print(f"Comparing with image: {filename}")
    known_image = cv2.imread(f"known_faces/{filename}", 0)
    if known_image is None:
      print(f"Failed to load image: {filename}")
      continue

    try:
      face_resized = cv2.resize(face_gray, known_image.shape[::-1])
    except Exception as e:
      print(f"Failed to resize face for comparison with {filename}: {e}")
      continue

    result = cv2.matchTemplate(known_image, face_resized, cv2.TM_CCOEFF_NORMED)
    _, score, _, _ = cv2.minMaxLoc(result)
    print(f"Match score with {filename}: {score:.2f}")

    if score > highest_score and score > 0.3:
      highest_score = score
      best_match_name = os.path.splitext(filename)[0]

  return best_match_name

# Create or connect to the database
connection = sqlite3.connect("face_detections.db")
cursor = connection.cursor()
cursor.execute("""
  CREATE TABLE IF NOT EXISTS face_log (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    image_name TEXT,
    person_name TEXT,
    x INTEGER,
    y INTEGER,
    w INTEGER,
    h INTEGER,
    detected_at TEXT
```

```
47        )
48    """)
49
50    # Request image path from user
51    image_path = input("Enter image path (e.g., test.jpg or C:/Users/...): ").strip()
52    image = cv2.imread(image_path)
52
53    if image is None:
54      print("Failed to load image. Please check the path.")
55      exit()
56
57    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
58
59    # Detect faces
60    face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_frontalface_default.xml')
61    faces = face_cascade.detectMultiScale(gray, scaleFactor=1.1, minNeighbors=5)
62
63    print(f"Number of detected faces: {len(faces)}")
64
65    # Process each detected face
66    for (x, y, w, h) in faces:
67      face_crop = gray[y:y+h, x:x+w]
68      name = recognize_face_by_template(face_crop)
69      print(f"Detected face at ({x},{y}) recognized as: {name}")
70
70      # Draw rectangle and label on the image
71      cv2.rectangle(image, (x, y), (x + w, y + h), (0, 255, 0), 2)
72      cv2.putText(image, name, (x, y + h + 25), cv2.FONT_HERSHEY_SIMPLEX, 0.8, (0, 255, 0), 2)
73
74      timestamp = datetime.now().strftime("%Y-%m-%d %H:%M:%S")
75      cursor.execute("""
76        INSERT INTO face_log (image_name, person_name, x, y, w, h, detected_at)
77        VALUES (?, ?, ?, ?, ?, ?, ?)
78      """, (os.path.basename(image_path), name, x, y, w, h, timestamp))
79
80    # Commit and close the database
81    connection.commit()
82    connection.close()
83
84    print("Face data saved to database.")
85    cv2.imshow("Identified Faces", image)
86    cv2.waitKey(0)
87    cv2.destroyAllWindows()
88
```

ما الذي يحدث عند تشغيل البرنامج

١. طلب إدخال مسار الصورة من المستخدم لمعالجتها.

٢. تحميل الصورة المحددة وتحويلها إلى تدرج رمادي Grayscale لتسهيل المعالجة.

٣. قراءة صور الوجوه المعروفة من مجلد known_faces كمرجع للتعرف على الوجه.

٤. اكتشاف الوجوه في الصورة باستخدام خوارزمية Haar Cascade من OpenCV.

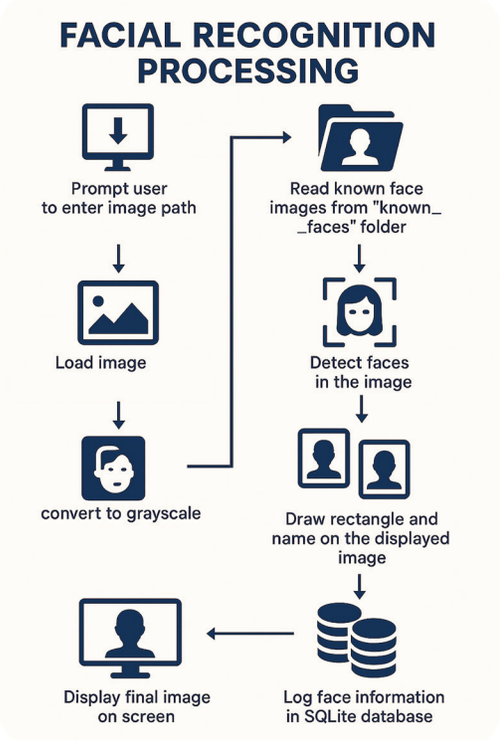
٥. مقارنة كل وجه مُكتشف مع الصور المرجعية باستخدام template matching لتحديد هوية الشخص.

٦. رسم مستطيل حول الوجه وكتابة الاسم عليه داخل الصورة المعروضة.

٧. تسجيل معلومات الوجه في قاعدة بيانات SQLite، مثل:

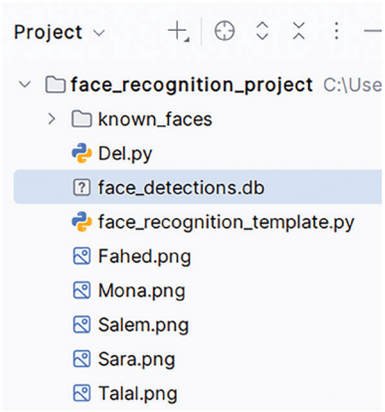
- اسم الصورة.
- اسم الشخص.
- موقع الوجه (x, y, w, h).
- وقت الكشف عن الوجه.

٨. عرض الصورة النهائية على الشاشة مع التعديلات.



معاينة قاعدة البيانات في برنامج DB Browser for SQLite

بعد تنفيذ البرنامج، يتم إنشاء قاعدة بيانات باسم face_detections.db في نفس مجلد المشروع، حيث يُمكن معاينة ملفات المشروع بالكامل من خلال Project Panel من برنامج PyCharm.

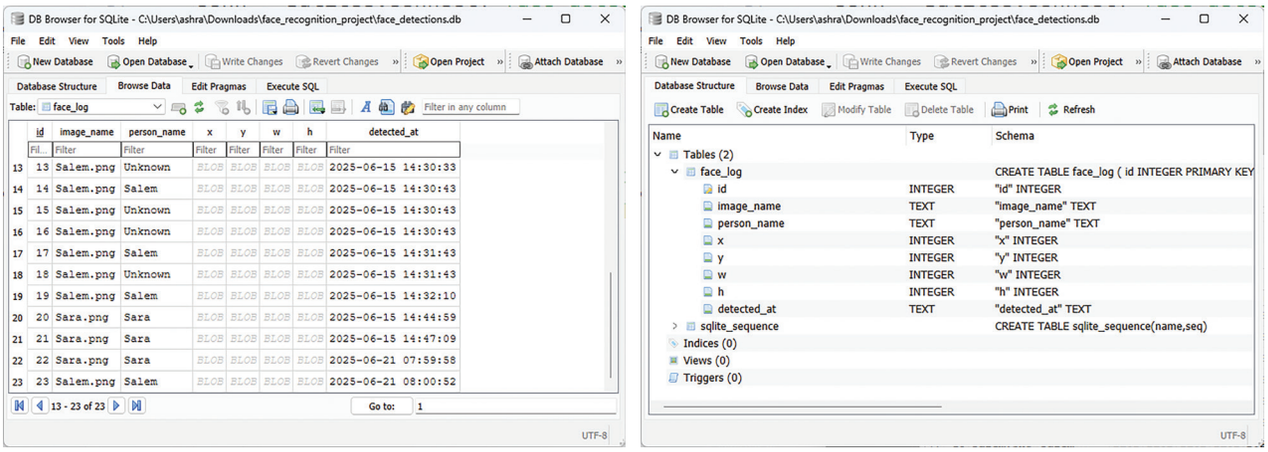


شكل رقم (8-1) يُوضح مكان إنشاء قاعدة البيانات

يُمكن استخدام برنامج DB Browser for SQLite لفتح قاعدة ومعاينة البيانات المسجلة فيها بسهولة عبر واجهة رسومية. كذلك يُمكن معاينة جدول حفظ البيانات حيث تظهر الأعمدة التالية:

- id: رقم الصف.
- image_name: اسم الصورة المستخدمة.
- person_name: اسم الشخص الذي تم التعرف عليه.
- x, y, w, h: إحداثيات وقياسات الوجه في الصورة.
- detected_at: تاريخ ووقت عملية التعرف على الوجه.

يُمكن استخدام هذه البيانات لأغراض مثل التوثيق، التحليل، تتبع الحضور.



شكل رقم (8-2) يُوضح معاينة جدول بيانات الوجوة التي تم التعرف عليها في برنامج DB Browser.

التقنيات المُستخدمة في المشروع

1 الفرق بين تقنية كشف الوجه وتقنية التعرف عليه.

كشف الوجه Face Detection:
عملية تحديد موقع وجود وجه داخل صورة أو فيديو، أي تحديد الإحداثيات (x, y)، والأبعاد الطول والعرض (h, w).

التعرف على الوجه Face Recognition:
عملية مقارنة الوجه المكتشف مع مجموعة من الوجوه المعروفة لتحديد هوية الشخص.

2 استخدامات مكتبة OpenCV / CV2 في هذا المشروع.

- قراءة الصور.
- تحويل الصور إلى رمادي.
- كشف الوجوه باستخدام كواشف مثل Haar Cascade.
- مطابقة الوجوه بواسطة matchTemplate.
- رسم المستطيلات وكتابة النصوص على الصور.
- عرض الصورة النهائية.

3 تحويل الصورة إلى تدرج رمادي.

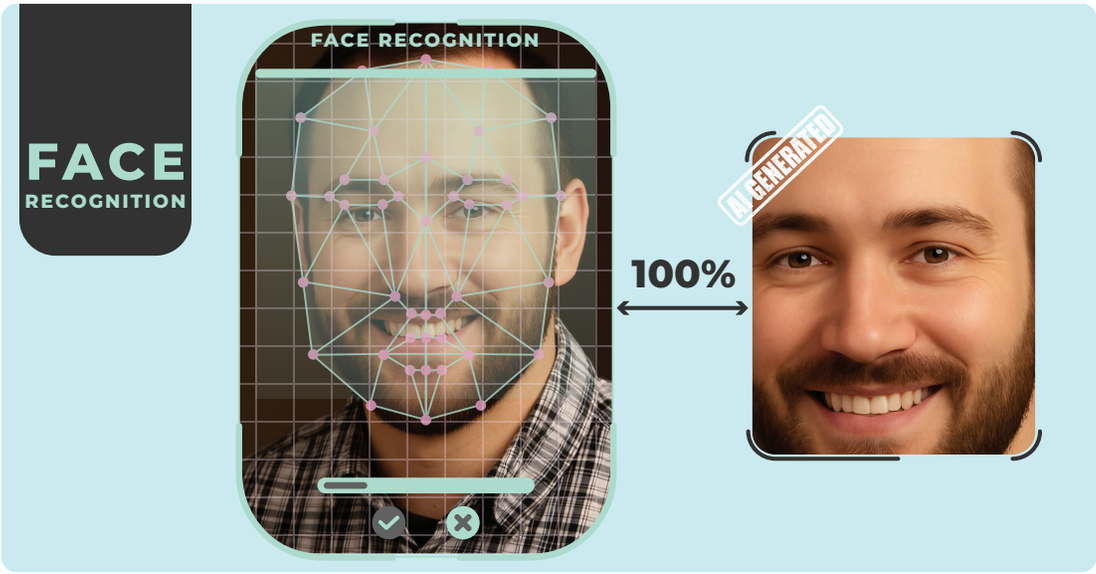
- يتم تحويل الصورة إلى تدرج رمادي لأن خوارزميات كشف الوجوه والتعرف عليها تعمل بشكل أسرع وأكفأ على الصور الرمادية (ذات القناة الواحدة) بدلاً من الصور الملونة (ذات 3 قنوات).
- تقليل حجم البيانات المراد معالجتها.
- معظم الكواشف والتقنيات تعتمد على تباين الإضاءة وليس الألوان.

4 فائدة استخدام قاعدة بيانات بدلًا من ملف نصي.

- تنظيم البيانات: تسهل التعامل مع بيانات كثيرة ومنظمة (مثل اسم الوجه، الوقت، المكان).
- البحث والاسترجاع السريع: يمكن استخدام استعلامات SQL للبحث والتحليل بسهولة.
- عدم التكرار: يمكن ضبط عبر القيود في الجداول.
- الاعتمادية والأمان: أكثر أمانًا ودقة في الحفظ والاسترجاع مقارنة بملف نصي.

5 فائدة استخدام matchTemplate.

- matchTemplate هي دالة من OpenCV تقوم بمقارنة صورة صغيرة (الوجه المُكتشف) مع صورة كبيرة (الوجه المعروف).
- تعيد (درجة التشابه) بين الصورتين كنسبة، ومن خلالها نحدد إذا كان الوجه مطابقًا أم لا.



المنتجات الرقمية

مراحل تصميم وتطوير المنتج الرقمي

Designing and Developing a Digital Product

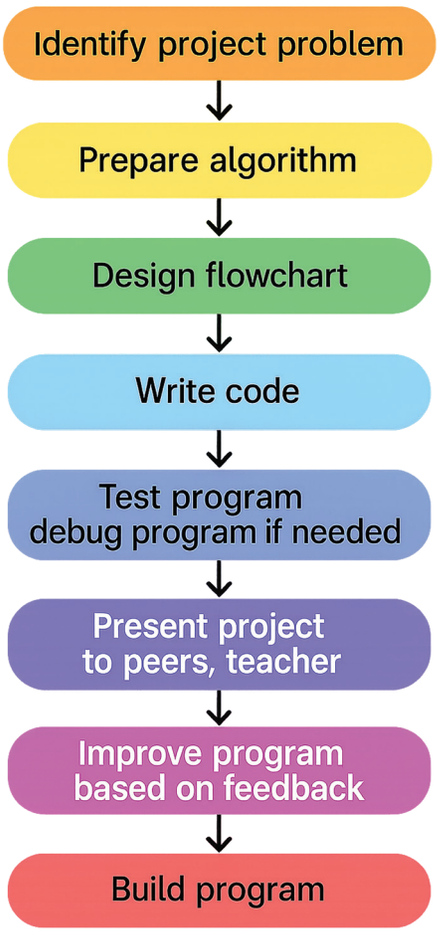
نواتج التعـلم

- توظيف مهارات التصميم والبرمجة لحل مشكلة واقعية بطريقة إبداعية.
- التعاون ضمن فريق لتوزيع المهام وتنفيذ المشروع حسب خطة مُنظمة.
- توثيق مراحل المشروع، ويعرضه على المعلم والزملاء مع التغذية الراجعة.

وحدة المشروعات

إن الهدف الأساسي من إنتاج المشروع هو الاستفادة من المهارات التي تم دراستها وتطبيقها من خلال أوراق العمل بالإضافة إلى تنمية مهارات العمل الجماعي التعاوني، والقدرة على تجميع الوسائط اللازمة لإنتاخ المشروع وفق أُسس تحليل النظم والبرمجة والقدرة على تنمية الابتكار.

من خلال دراستك لوحدة البرمجة بلغة Python، والاطلاع على تطبيقاتها المتعددة، والمساعدة المتوافرة في رمز الاستجابة السريع QR في بداية وحدة المُنتجات الرقمية، صمم المشروع الخاص بك، متبعًا الخطوات التالية:



- تحديد مشكلة المشروع.
- إعداد الخوارزمية.
- تصميم خريطة التدفق.
- كتابة التعليمات البرمجية.
- اختبار البرنامج، تصحيح البرنامج إن وجدت.
- عرض المشروع على زملائك، ومعلمك.
- تحسين البرنامج وفق التغذية الراجعة.
- بناء البرنامج.

ضوابط وإرشادات تنفيذ المشروع

سنستعرض الآن طريقة إعداد المشاريع ضمن ضوابط وإرشادات محددة لتنسيق العمل باتباع الخطوات التالية:

مخطط إعداد مشروع في PyCharm

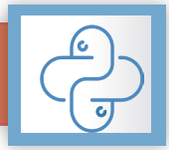
مهام فريق العمل

يتكون الفريق من 4 إلى 5 أعضاء، تُقسم المهام بينهم على النحو التالي:

- قائد الفريق: المسؤول عن إدارة الفريق وتجميع الملفات والمستندات المطلوبة وتقديمها للمعلم.
- مُعد الخوارزمية: المسؤول عن إعداد الخطوات الأساسية لعمل المشروع.
- مصمم خريطة التدفق: المسؤول عن ترجمة الخوارزمية إلى خريطة التدفق.
- المبرمج: المسؤول عن ترجمة خريطة التدفق لتعليمات برمجية واختبارها والتأكد من سلامتها.
- مُعد العرض التقديمي: المسؤول عن إعداد وتصميم العرض التقديمي للمناقشة.

تُقسم المهام بين أعضاء الفريق على النحو التالي:

اسم المتعلم	المهمة	الوظيفة
	قائد الفريق	المسؤول عن إدارة عمل الفريق وتجميع الملفات والمستندات المطلوبة وتقديمها للمعلم



مهام فريق العمل

الاسبوع	المرحلة	الناتج المُتوقع
الأول	التخطيط والتحليل	☐ اختيار فكرة المشروع.
		☐ تحديد المشكلة والأهداف.
		☐ إعداد وثيقة المتطلبات الأولية.
الثاني	خطة المشروع والتوزيع	☐ توزيع المهام على أعضاء الفريق.
		☐ إعداد خطة تنفيذ المشروع أسبوعياً.
الثالث	التصميم والتوثيق الأولي	☐ تصميم الواجهة.
		☐ إعداد خطة العمل بالتفصيل.
الرابع	البرمجة والتنفيذ الأولي	☐ البدء في تنفيذ المشروع عمليًا.
		☐ تجربة الأدوات البرمجية والتقنية.
الخامس	الاختبار والتحسين	☐ مراجعة الأخطاء البرمجية.
		☐ اختبار الأداء.
		☐ إجراء التعديلات والتحسينات النهائية.
السادس	**التوثيق النهائي والعرض**	☐ كتابة التقرير النهائي.
		☐ إعداد العرض التقديمي.
		☐ التدريب على العرض وتقديمه أمام الصف.

ملاحظة: يختلف عدد أسابيع المشروع والمراحل المختلفة وفق خطة توزيع المنهج.

فكرة المشروع

تصميم مشـــــروع تقني تعليمي بســـــيط لحل مشـــــكلة أو تنفيذ فكرة مفيدة بطريقـــــة إبداعية على أن تكون واضحـــــة وقابلة للتطبيق.

يناقش أعضاء الفريق ويحدد موضوع يثير اهتمامهم، سواءً كان تطبيقاً أو لعبة أو أداة تحليل البيانات وغيرها، على أن تكون مُطابقة للشروط التالية:

- أن تكون الفكرة واضحة وقابلة للتطبيق مع شرح المشكلة والحلول.
- تحديد الفئة المستفادة.
- أن تكون الفكرة قابلة لتصميم نموذج برمجي.
- أن تحتوي على عدة مصادر خارجية يسهل الرجوع إليها.
- أن تعتمد على تقنية اكتشاف الوجوه أو أي تقنية من تقنيات الذكاء الاصطناعي.

سجل فكرة المشروع بعد مناقشة أعضاء الفريق:

أهداف المشروع

يهدف المشـــــروع إلى إكســـــاب المتعلم مهـــــارات تقنية وإبداعيـــــة عبر تصميم وتنفيـــــذ فكرة تعليمية بطريقة منظمـــــة وتعاونية.

سجل أهداف مشروعك:

التوظيف الواقعي في الحياة العملية

يساهم المشروع في ربط التقنية بواقع الحياة من خلال توظيف البرمجة والذكاء الاصطناعي وقواعد البيانات في حل مشكلات حقيقية أو تحسين ممارسات يومية بطريقة ابتكارية.

سجل كيف ستوظف المشروع الخاص بك في الحياة العملية:

مهارات المشروع

يهدف المشروع إلى تنمية مهارات المتعلمين التقنية مثل التصميم والبرمجة، إلى جانب المهارات الشخصية كالتعاون وحل المشكلات، وذلك من خلال تنفيذ فكرة عملية لإكسابهم خبرات واقعية تعزز جاهزيتهم للمستقبل، وتساعدهم على اكتساب كفاءات تقنية تعزز قدرتهم على الابتكار والتطبيق العملي.

حدد المهارات المستخدمة بالمشروع:

	المجال	المهارات الفرعية المستهدفة
☐	**المهارات الرقمية والتقنية**	☐ القدرة على كتابة أو تعديل أكواد برمجية بلغة Python.
		☐ استخدام التقنيات المناسبة لقراءة الوجوه والتعرف عليها تلقائيًا.
		☐ تخزين ومعالجة البيانات باستخدام قاعدة بيانات SQLite.
		☐ معرفة كيفية حفظ الصور والتعامل مع ملفات الوجوه المخزنة.
☐	**التفكير الناقد وحل المشكلات**	☐ تحليل المشكلة وتحديد عناصرها.
		☐ تجريب الحلول واختيار الأنسب.
		☐ التحقق من دقة النتائج وتعديل الأخطاء.
☐	**مهارات التوثيق والبحث العلمي**	☐ تصوير مراحل تنفيذ المشروع وتسجيلها.
		☐ جمع معلومات من مصادر موثوقة.
		☐ كتابة المراجع العلمية.
☐	**مهارات العمل الجماعي**	☐ التعاون وتقسيم المهام.
		☐ تحمل المسؤولية والالتزام بالوقت.
		☐ اتخاذ قرارات جماعية.
☐	**مهارات العرض والتواصل**	☐ إعداد عرض بصري فعّال (شرائح، فيديو، أدلة، تقارير).
		☐ شرح الفكرة بلغة واضحة ومقنعة.
		☐ التفاعل مع الجمهور والرد على الأسئلة.

خطوات تنفيذ المشروع

1. إعداد بيئة العمل

تثبيت ما يحتاج إليه الفريق من برمجيات وأدوات لإعداد المشروع، ومنها:

- PyCharm: لتصميم النموذج البرمجي واستيراد المكاتب اللازمة.
- PowerPoint: لإعداد العرض التقديمي.
- Windows screen recorder: لتسجيل الشاشة إستعداداً لتوثيق المشروع.
- Draw.io: لرسم خريطة التدفق للمشروع.
- Microsoft Edge: للبحث عن المعلومات.

و أي مستلزمات أخرى تخدم المشروع.

2. إعداد هيكل المشروع

يُفضل الاستغلال الأمثل للمهارات البرمجية التالية:

- استخدام لغة Python.
- استخدام أنواع متعددة من المتغيرات.
- استخدام التعليمات البرمجية للشروط Conditions.
- معالجة الأخطاء باستخدام الاستثناءات try/ except.

3. كتابة التعليمات البرمجية

يكتب المبرمج التعليمات البرمجية استناداً لما تم تخطيطه سابقاً.

4. اختبار المشروع

يختبر المبرمج البرنامج، ويتأكد من خلوه من الأخطاء البرمجية، اللغوية، والعلمية.

5. تطوير المشروع

يُبسط المبرمج ويرتب التعليمات البرمجية إذا أصبحت صعبة القراءة، وإضافة التعليقات والبيانات الإرشادية التي توضح مهمة التعليمات البرمجية.

6. توثيق المشروع

يُنشئ مصمم العرض التقديمي عرضًا شاملاً للجوانب التالية:

- عرض بيانات المدرسة وأعضاء الفريق.
- الترحيب بالمعلم وزملائه المتعلمين.
- عرض مقدمة عن فكرة المشروع موضحاً الهدف والمشكلة والفئة المستفادة والحل.
- عرض خريطة التدفق.
- عرض النموذج البرمجي.
- عرض المصادر.
- فتح باب الأسئلة والمناقشة.
- الختام.

و تسليم مجلد مجمع لجميع ملفات المشروع.

7. مشاركة المشروع

عرض الفريق للمشروع، ومناقشته مع المعلم والمتعلمين.

8. الاستمرار في التعلم

التوسع في الاطلاع على مستجدات الذكاء الاصطناعي وتنمية قدرات المتعلم البرمجية من خلال الدورات التدريبية والمنتديات الحاسوبية.

التوثيق العلمي للمشروع

- المراجع.
  - الالتزام بطريقة علمية لتوثيق المراجع على سبيل المثال نظام APA[1].
  - عرض ومشاركة المشروع.
- يقوم الفريق بعرض المشروع ومناقشته مع المعلم والمتعلمين، مع مشاركته على منصة Teams.
- الاستمرار في التعلم.

مشاريع إثرائية

1 اختصار لعبارة American Psychological Association

ويُعد أحد أشهر أنظمة التوثيق المرجعي المستخدمة عالميًا، خاصةً في الأبحاث العلمية، والعلوم الاجتماعية، والنفسية، والتربوية، والبحث الأكاديمي عمومًا.

المراجع

- مؤسسة بايثون للبرمجيات. (2024). دليل بايثون الرسمي: إصدار 7. https://www.python.org/doc.
- منظمة الأمم المتحدة للتربية والعلم والثقافة، « الذكاء الاصطناعي في التعليم» (2017).
- خارطة الطريق للتحول الرقمي للمؤسسات الحكومية في دولة الكويت، د. عبد الله محمد عبد الكريم المطوع، مركز دراسات الخليج والجزيرة العربية. العدد 32 . 2023م.
- التوصية الخاصة بأخلاقيات الذكاء الاصطناعي- منظمة الأمم المتحدة للتربية والعلم والثقافة (اليونسكو)، 2021م.
- محمد لحلح. (2020). مدخل الى الذكاء الاصطناعي وتعلم الآلة. شركة حسوب وأكاديمية حسوب.
- نرمين مجدي. (2020). الذكاء الاصطناعي وتعلم الآلة. صندوق النقد العربي.
- منظمة الأمم المتحدة للتربية والعلم والثقافة. (2021). الذكاء الاصطناعي والتعليم. منظمة الأمم المتحدة للتربية والعلم والثقافة.
- سدايا. (سبتمبر 2023). مبادئ أخلاقيات الذكاء الاصطناعي. سدايا.
- Basu, S. (2021, May 20). Learn SQLite with Python in 24 Hours: Simple, Concise & Easy Guide to Using Database with Python. Independently published.
- Kurniawan, A. (2021, January 30). Python and SQLite Development. PE Press.
- Siahaan, V. S., & Sianipar, R. H. (2025). Learn SQLite with Python: Building Database-Driven DesKtop Projects. Independently published.
- Hayes, W. (2025, 1 فبراير). Master Computer Vision and AI with OpenCV and Python: UnlocK Cutting-Edge Tools, Techniques, and Real-World Applications for Image Processing and Machine Learning. Independently published.
- Howse, J., & Minichino, J. (2020, 20 فبراير). Learning OpenCV 4 Computer Vision with Python 3 (3rd ed.). PacKt Publishing.
- Mugesh, S. (2023). Hands-on ML Projects with OpenCV: Master computer vision and Machine Learning using OpenCV and Python. Independently published.

11