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

أ.منى سالم عوض سالم (رئيس اللجنة)

تأليف

أ.رأفـت صابـر عبـدالله أحمـــد

أ.حســـــام الدين علي عبـد القادر

د.أشرف رضوان رضـوان سليمـان

د. يوسف منصـور يوسف الخليفي

أ.منى مرزوق مخلد العـــــــازمي

أ.مـنار مصطفى عبدالحميد جمال

أ.إبراهيم عبدالله إبراهيم المياس

تصميم

أ.ســـاره ياســـين عبدالله الأمـير

إخراج

د.أشرف رضوان رضـوان سليمـان

الطبعة الثانية
1448 هـ
2027/2026 م

حقوق التأليف والطبع والنشر محفوظة لوزارة التربية - الإدارة العامة للتواجيه والبحوث والمناهج

إدارة البحوث التربوية والمناهج

الطبعة الأولى 2026/2025م

الطبعة الثانية 2027/2026م

لجنــة المواءمة

إشــراف

أ.منى سالم عوض سالم (رئيس اللجنة)

أ.رأفــت صابــر عبــداللاه أحمـــد

أ.حســام الدين علي عبــد القـــادر

د.أشرف رضوان رضوان سليــمان

أ.مــنار مصطفى عبدالحميد جــمال

أ.إبراهيم عبدالله إبراهيم المياس

تصميم

أ.ســـاره ياســـين عبدالله الأمــير

إخــراج

د.أشرف رضوان رضــوان سليمــان

الفريق المُساند

أ.أحمــد محمــد عبداللــه عيســى

د. عمـــرو محمــود إبراهيم حبيب

تصميم

أ. سنيـــــه محمد علــي المؤمــن

المراجعة العلمية

د. أشرف رضـــوان رضـــوان سليمـــان

أ. فـــاطمة نجـــم جـــاسم الهـــولي

أ. عبدالرحمن محمد مال الله الجزاف

المراجعة اللغوية

أ. عـــبد الرازق محـــمد عصـــفور

أودع في مكتبة الكويت الوطنية - ردمك : 978-9921-33-019-9

أودع في مكتبة وزارة التربية رقم (٥٥) بتاريخ ٢/ ٩/ ٢٠٢٥م

مطبعة حكومة دولة الكويت

Government Press - State of Kuwait

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

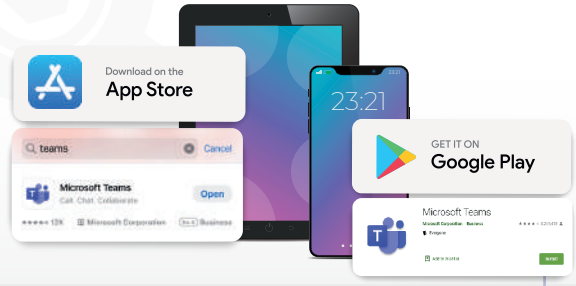
سمو الشيخ صباح خالد الحمد الصباح
ولي عهد دولة الكويت

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.

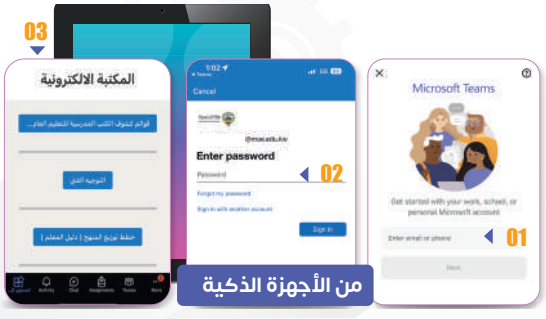
ثانياً: تفعيل Microsoft Teams على الجهاز الرقمي

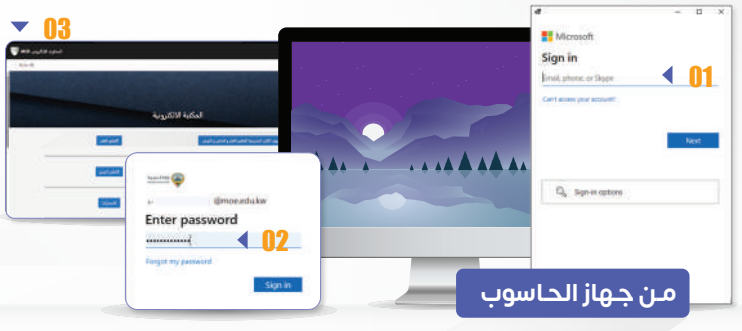
لتشغيل التطبيق اتبع الخطوات التالية:

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 وتنجح بامتياز.

مع حـمد Chat

مـا هـي خـدمـة حـمـد شـات؟

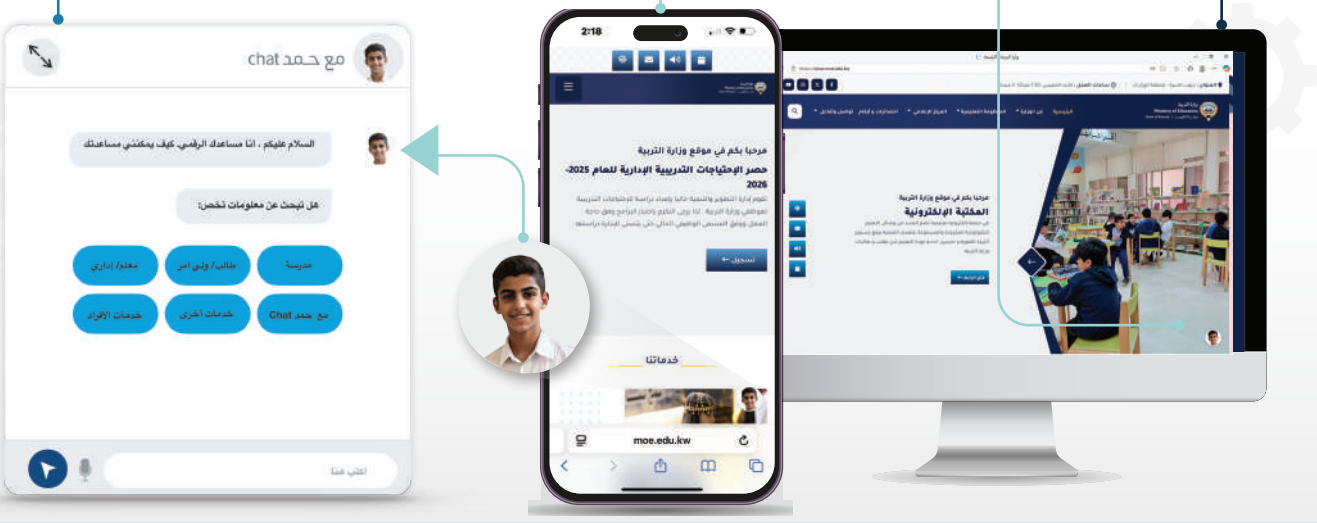
هـي خدمـة المحادثـة الذكيـة المقدمـة مـن وزارة التربيـة فـي دولـة الكويـت، تعتمـد علـى الـذكاء الاصطـناعي، تتمثـل وظيفتـه الأساسـية فـي تسـهيل عمليـة الفهـم والاسـتيعاب مـن خلال تقديـم الشـروحات المبسـطة، والإجابـة علـى الاستفسـارات، وتوجيـه المتعلميـن خلال مسـيرتهم التعليميـة.

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	13
الوحدة الأولى: قواعد البيانات SQLite في Python	15
- مدخل إلى قواعد البيانات Introduction to Database	17
- إنشاء قواعد البيانات Creating a Database	31
- إنشاء الجداول Creating Tables	41
- إضافة البيانات Inserting Data	53
- الاستعلام عن البيانات Querying Data	67
- تحديث البيانات Updating Data	83
- حذف البيانات Deleting Data	95
الوحدة الثانية: المنتجات الرقمية Digital Products	107
- الذكاء الاصطناعي وتكامله مع قواعد البيانات AI and Database Integration	109
- مراحل تصميم وتطوير المنتج الرقمي Designing and Developing a Digital Product	125
المراجع	138

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 وحدة 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 الحديث، أصبحت البيانات عنصرًا أساسيًا ومحركًا رئيسيًا للتطور واتخاذ القرارات، مما أدى إلى الحاجة إلى وسائل فعّالة لتنظيمها وتخزينها وإدارتها بكفاءة. ومن هنا برزت أهمية قواعد البيانات، التي تؤدي دورًا محوريًا في تنظيم المعلومات وهيكلتها وتسهيل الوصول إليها ومعالجتها بسرعة ودقة وموثوقية.

وتُعد قواعد البيانات أساسًا للأنظمة والتطبيقات الحديثة، حيث تُسهم في إدارة المعلومات وتحليل البيانات ودعم اتخاذ القرار في مختلف المجالات. لذلك، فإن إتقان قواعد البيانات يُمثل خطوة مهمة لفهم التكنولوجيا الحديثة وتطوير تطبيقات أكثر ذكاءً وكفاءة.

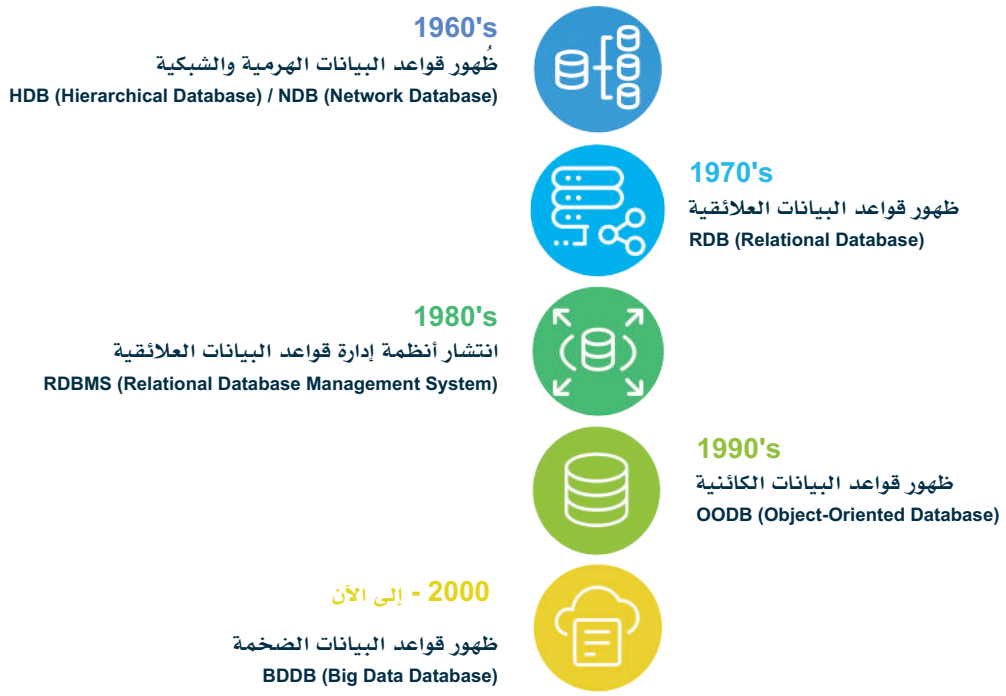
أهمية قواعد البيانات

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:

شكل رقم (1-1) يمثل أهمية قواعد البيانات.

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 المدعومة بالذكاء الاصطناعي AI-Powered Big Data DBs جزءًا أساسيًا من التحول الرقمي والخدمات السحابية.



شكل رقم (1-2) يمثل التطور التاريخي لقواعد البيانات.

مميزات قواعد البيانات

شكل رقم (1-3) يمثل أهم مميزات قواعد البيانات.

استخدامات قواعد البيانات في الحياة اليومية

شكل رقم (1-4) أمثلة على استخدامات قواعد البيانات.

الجوانب الأخلاقية الواجب مراعاتها عند التعامل مع قواعد البيانات

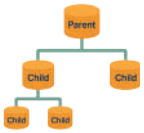
عند التعامل مع قواعد البيانات، من الضروري الالتزام بأخلاقيات تقنية المعلومات لحماية البيانات وضمان سلامتها، ومن أبرز الجوانب الأخلاقية ما يلي:

- 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

هناك عدة أنواع من قواعد البيانات، وكل نوع يُستخدم حسب طبيعة البيانات وطريقة تنظيمها، من أبرزها:

- قواعد البيانات الهرمية Hierarchical Databases:



تشبه شكل الشجرة، حيث ترتبط البيانات بطريقة "الأب Parent - الابن Child". كانت تُستخدم في أنظمة الحاسوب القديمة مثل IMS من IBM.

- قواعد البيانات الشبكية Network Databases:

هي نوع من قواعد البيانات الهيكلية تُنظّم فيها السجلات Records على شكل عُقد Nodes مترابطة عبر مؤشرات Pointers، مما يسمح بتمثيل العلاقات متعددة إلى متعددة Many-to-Many بشكل مباشر.

- قواعد البيانات العلائقية Relational Databases:

تعتمد على الجداول والعلاقات، وهي الأكثر استخدامًا حاليًا، مثل: Oracle، MySQL، SQLite، وهي النوع الذي سنتعرض له في هذا الكتاب.

- قواعد البيانات الكائنية Object-Oriented Databases:

تُستخدم لتخزين البيانات في شكل كائنات Objects المستخدم في البرمجة الكائنية التوجه OOP، مثل لغة Java أو C++.

- قواعد بيانات NoSQL:

تتعامل مع كميات ضخمة من البيانات غير المنظمة، وتُستخدم في تطبيقات الويب والموبايل مثل MongoDB، وFirebase.

- قواعد البيانات الموزعة Distributed Databases:

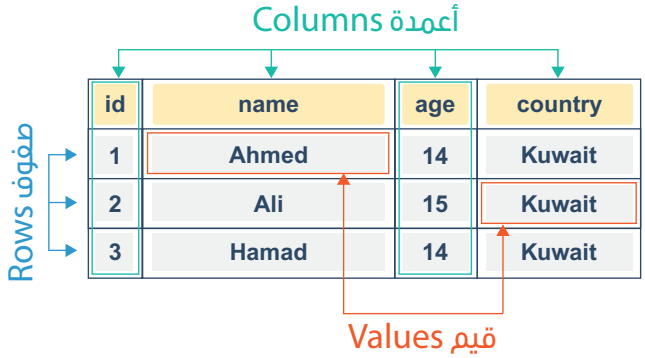
تكون البيانات مخزنة على أكثر من خادم Server في أماكن مختلفة، لكنها تُدار كأنها قاعدة واحدة.

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 يُمثل الجدول كياناً Entity معيناً (مثل جداول: بيانات الطلاب، الكتب، معالم إحدى الدول)، ويتكون الجدول من أعمدة وصفوف.



شكل رقم (1-5) يوضح مُكونات جدول بقاعدة بيانات.

2. الأعمدة Columns:

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

3. الصفوف Rows:

الصف هو وحدة بيانات أفقية في جدول قاعدة البيانات، تحتوي على مجموعة من القيم المرتبطة بخصائص محددة؛ يُمثل نسخة واحدة Instance من الكيان الذي مثله الجدول.

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 في الجدول، تُشير إلى عمود أو مجموعة أعمدة (مفتاح أساسي) في جدول آخر، ويُستخدم لربط الجداول ببعضها.

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).

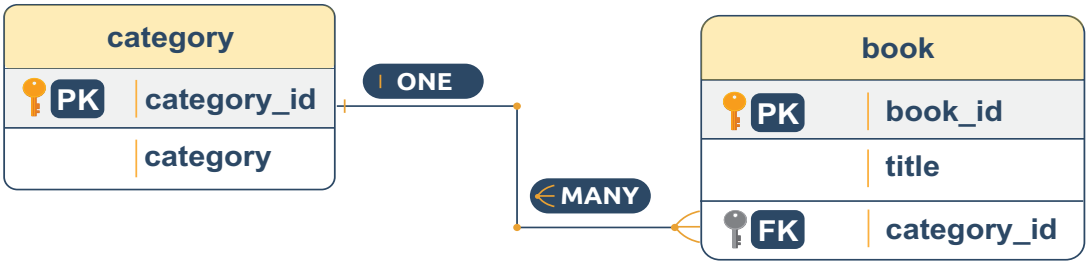
أنواع العلاقات بين الجداول

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

- واحد إلى واحد One to One: العلاقة التي يُمكن أن يرتبط فيها كل صف في الجدول الأول (الرئيسي) بصف واحد فقط في الجدول الثاني (الفرعي)، بينما يرتبط كل صف في الجدول الفرعي بصف واحد فقط بالجدول.

شكل رقم (1-6) يُوضح مخطط العلاقة من نوع واحد إلى واحد.

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



شكل رقم (1-7) يُوضح مخطط العلاقة من نوع واحد إلى مُتعدد.

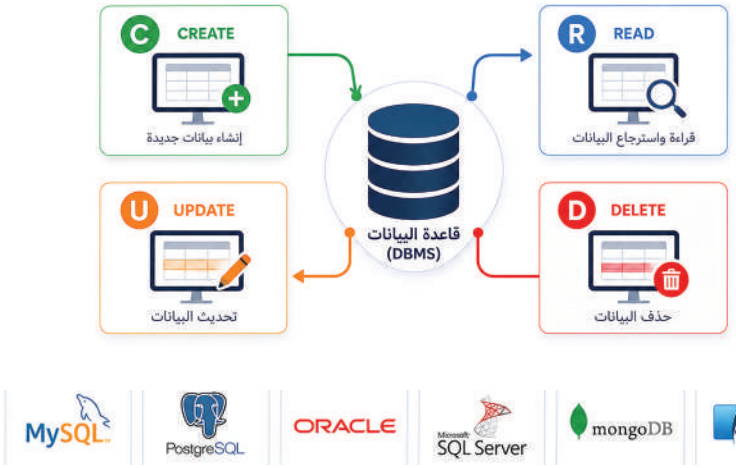
- **متعدد إلى متعدد Many to Many: العلاقة التي يُمكن أن يرتبط فيها كل صف في الجدول الأول بعدة صفوف في الجدول الثاني، والعكس صحيح، ولا يمكن الربط بين الجدولين بشكل مباشر، بل يتم إنشاء جدول ربط Join Table بين الجدولين المرتبطين لتمثيل هذه العلاقة.**

شكل رقم (1-8) يُوضح مخطط العلاقة بين جدولين مرتبطين بعلاقة متعدد إلى متعدد، وجدول الربط بينهما.

في الشكل السابق يوضح أن الكتاب يُمكن أن يكون له عدة مؤلفين، والمؤلف يُمكن أن يكون له عدة كتب. هذه العلاقة تُسمى Many to Many، وتم تمثيلها بإنشاء جدول ربط من نوع (الجدول الوسيط Junction Table)[1] يُسمى author_book.

نظام إدارة قواعد البيانات (DBMS)

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



شكل رقم (1-9) يُمثل العمليات الأساسية في أنظمة إدارة قواعد البيانات CRUD.

[1] جدول يُستخدم لربط جدولين في علاقة متعدد إلى متعدد من خلال تخزين مفاتيحهما الخارجية فقط.

برنامج DB Browser for SQLite

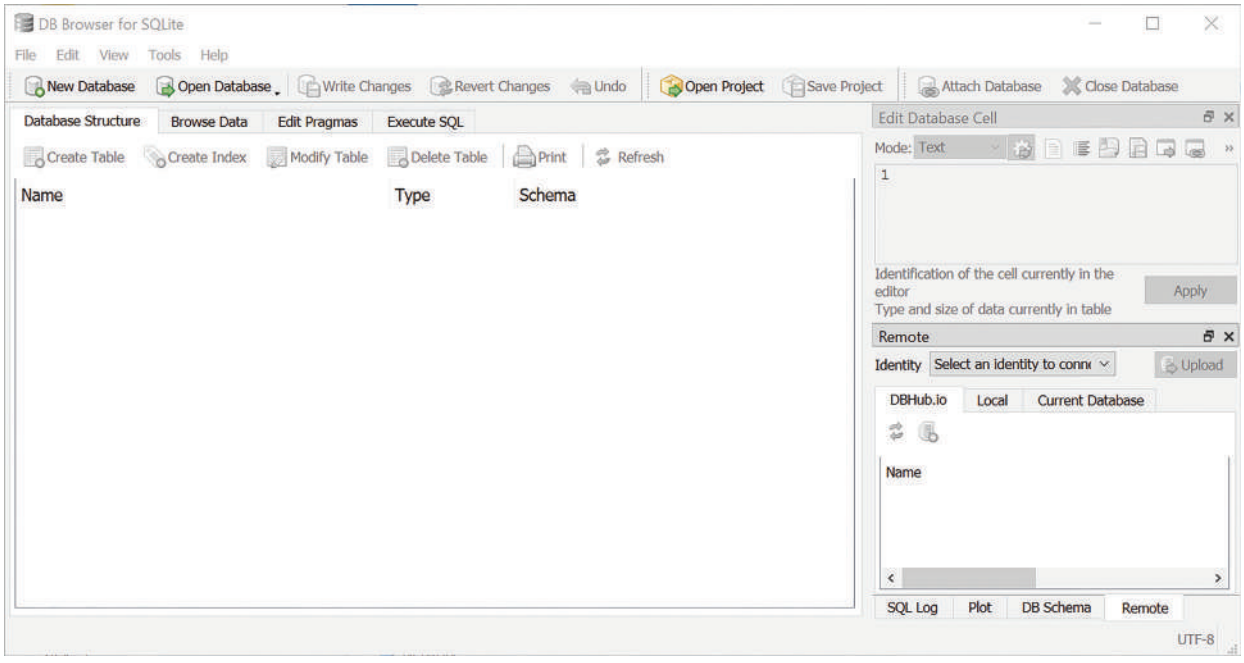
هو برنامج مجاني مفتوح المصدر، يُستخدم لإدارة قواعد البيانات من نوع SQLite، ويتميز بواجهته الرسومية البسيطة التي تساعد المستخدمين على إنشاء الجداول عبر إدخال وتعديل البيانات، وتنفيذ الاستعلامات المختلفة وعرض النتائج بطريقة مُنظمة تسهل قراءتها.

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 (1-10) 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.

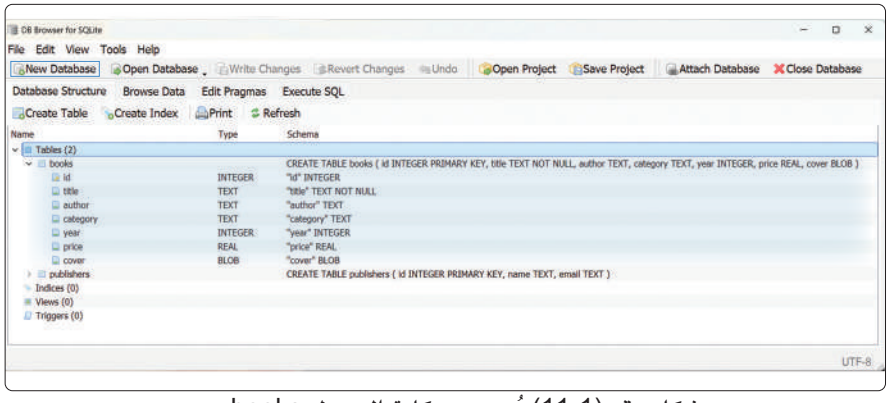
علامات التبويب الرئيسية Main Tabs

- 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 أوامر SQL.

التعامل مع برنامج DB Browser for SQLite

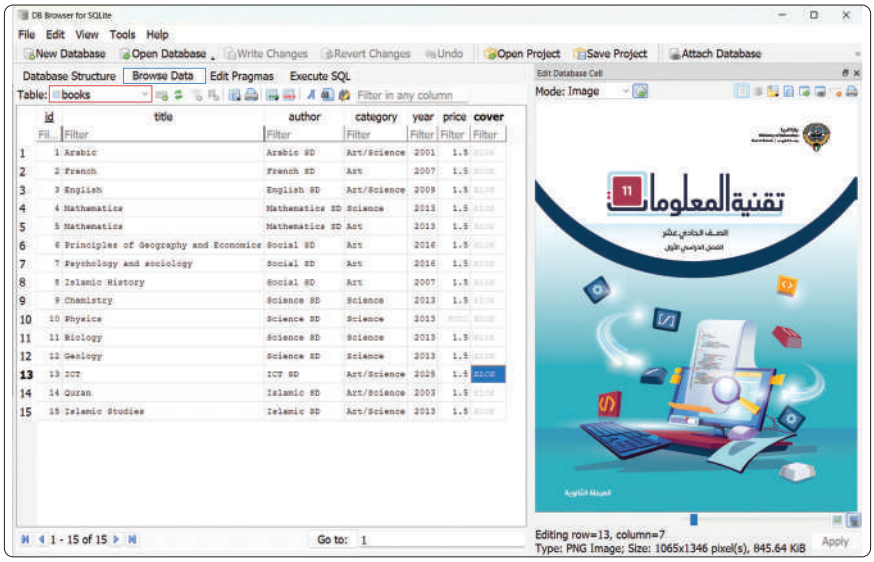
يُتيح البرنامج التعامل مع قواعد البيانات SQLite من خلال العديد من العمليات منها:

- 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 (1-11) يُوضح هيكلية الجدول books.

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



شكل رقم (1-12) 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، يجب إغلاقها بطريقة صحيحة 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) التاريخ:

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 landmarks 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 landmarks:
   - أسماء الأعمدة: .....
     .....
4. عاين بيانات الجدول landmarks، وتعرف على مكوناته ثم استكمل ما يلي:
   - موقع 'Al-Tahrir Tower' location: .....
   - سنة التأسيس 'The Avenues mall' year: .....
   - عَدَدْ 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 الوحدة القياسية sqlite3، دون الحاجة إلى خادم خارجي (Server).

صيغة التعليمة البرمجية للاتصال بقاعدة البيانات SQLite في Python

دالة من وحدة sqlite3
لإنشاء / فتح اتصال بقاعدة البيانات

كائن (مُتغير) الاتصال بقاعدة البيانات
Database Connection Object

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

وحدة قواعد البيانات

اسم قاعدة البيانات

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.
   - استيراد وحدة 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

- استيراد وحدة 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 وحدة sqlite3.

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

- إنشاء / فتح اتصال بقاعدة البيانات.
  - sqlite3.connect: تُستخدم دالة connect من وحدة 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()
```

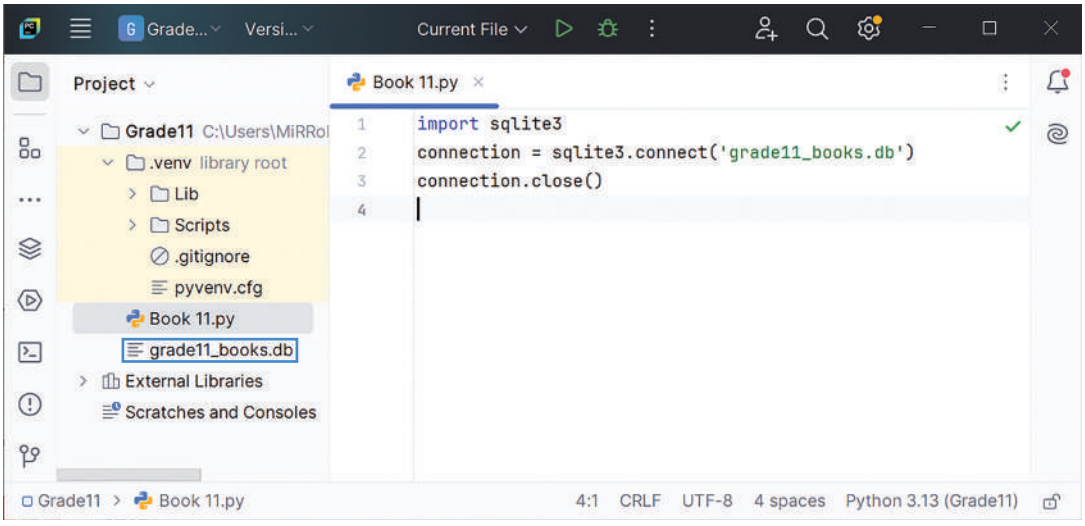
- إغلاق الاتصال بقاعدة البيانات.

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:
  - ضمان تحرير موارد الجهاز، خاصة الذاكرة.
  - ضمان سلامة البيانات وعدم فقدها.
  - 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 (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

- استيراد وحدة 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.')
12      else:
13          print('Failed to connect to the database.')
```

التفسير

```
import sqlite3
```

- استيراد وحدة 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.')
```

- طباعة رسالة تم إغلاق الاتصال.

```
else:
```

- في حال فشل الاتصال بقاعدة البيانات.

```
print('Failed to connect to the database.')
```

- طباعة رسالة فشل الاتصال بقاعدة البيانات.

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) التاريخ:

إنشاء قاعدة بيانات Kuwait_landmarks.db لتوثيق معالم دولة الكويت.

تطبيق الخطوات التالية:

1. أنشئ مشروعًا جديدًا باسم landmark_database، مستخدمًا بيئة التطوير المناسبة IDE.
2. أنشئ ملف Python جديد باسم data_database.py.
   - استورد وحدة 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 هيكلًا أساسيًا في قواعد البيانات، حيث تُستخدم لتخزين البيانات وتنظيمها وإجراء العمليات الأساسية عليها، والتي تتمثل في (الإدخال، والاستعلام، والتعديل، والحذف) بسهولة وكفاءة. ويتكون الجدول من مجموعة من الصفوف Rows والأعمدة Columns، بينما تُمثل الخلايا Cells مناطق التقاطع بين الصفوف والأعمدة، وتحتوي على القيم الفعلية للبيانات.

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

Column Name
Primary Key
Row
Value

شكل رقم (3-1) يوضح المُكونات الأساسية لجدول في قاعدة البيانات.

أنواع البيانات Data Types في SQLite

هي تصنيفات تُحدد نوع وطبيعة البيانات التي يمكن تخزينها في أعمدة الجداول داخل قاعدة البيانات، حيث تُسهم في تنظيم البيانات وضمان تخزينها ومعالجتها واسترجاعها بطريقة صحيحة وفعّالة.

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	تُشير إلى غياب القيمة.	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_name 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 (2-3) 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)
```

- PRIMARY KEY من نوع INTEGER يزيد القيمة تلقائيًا بمقدار واحد.
- AUTOINCREMENT: سلوك خاص يُستخدم مع المفتاح الأساسي PRIMARY KEY من النوع INTEGER، لزيادة القيمة تلقائيًا بمقدار واحد لكل صف جديد، مع ضمان عدم إعادة استخدام القيم التي تم حذفها.
- 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

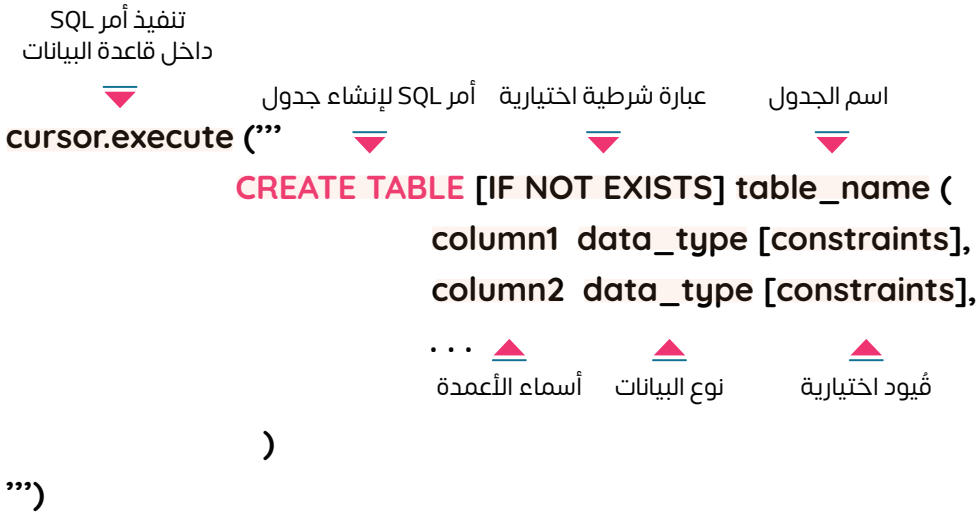
يجب عند إنشاء جدول تحديد التالي:

- **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

- يُسمح باستخدام الأحرف (A-Z, a-z) والأرقام (0-9) والشرطة السفلية (_).
- يجب أن يبدأ الاسم بحرف أو بشرطة سفلية، ولا يجوز أن يبدأ برقم.
- تجنب استخدام المسافات أو الرموز الخاصة.
- يجب تحقيق خاصية التفرّد Uniqueness، بحيث يكون اسم كل جدول فريدًا داخل قاعدة البيانات، واسم كل عمود فريدًا داخل الجدول نفسه.
- يُنصح باستخدام صيغة الجمع Plural عند تسمية الجداول.
- يجب عدم استخدام الكلمات المحجوزة Reserved Words.
- يُفضّل اختيار أسماء واضحة ومعبرة تعكس محتوى الجدول أو العمود.
- يُفضّل الالتزام بأسلوب تسمية موحّد Consistency مثل snake_case التي تستبدل المسافة بين الكلمات بشرطة سفلية.

صيغة التعليمة البرمجية لإنشاء جدول في قاعدة البيانات SQLite في Python



شكل رقم (2-3) يوضح صيغة كتابة التعليمة البرمجية لإنشاء جدول[1].

1 أسماء الجداول والأعمدة غير حساسة لحالة الأحرف Case-Insensitive حروف كبيرة وصغيرة.

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.

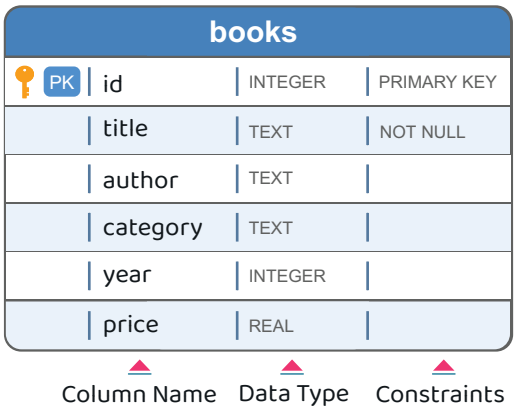
خطوات إنشاء جدول في قاعدة البيانات

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:
   - استيراد وحدة 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 for SQLite.

إنشاء جدول في قاعدة البيانات

مثال 1 : إنشاء جدول books في قاعدة البيانات grade11_books

الشكل التالي يُوضح أسماء الأعمدة، أنواع البيانات، والقيود لهيكلية الجدول المطلوب إنشاؤه.



شكل (3-3) يُمثل هيكلية جدول بقاعدة البيانات.

خطوات التنفيذ

- استيراد وحدة 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 for SQLite.

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

التفسير

import sqlite3

- استيراد وحدة sqlite3.

---

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

- الاتصال بقاعدة بيانات grade11_books.db، فإذا كانت قاعدة البيانات غير موجودة، فسيتم إنشاؤها تلقائيًا.

---

cursor = connection.cursor()

- إنشاء كائن المؤشر cursor.

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

- إنشاء الجدول books.

```
connection.commit()
```

- حفظ التغييرات.

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

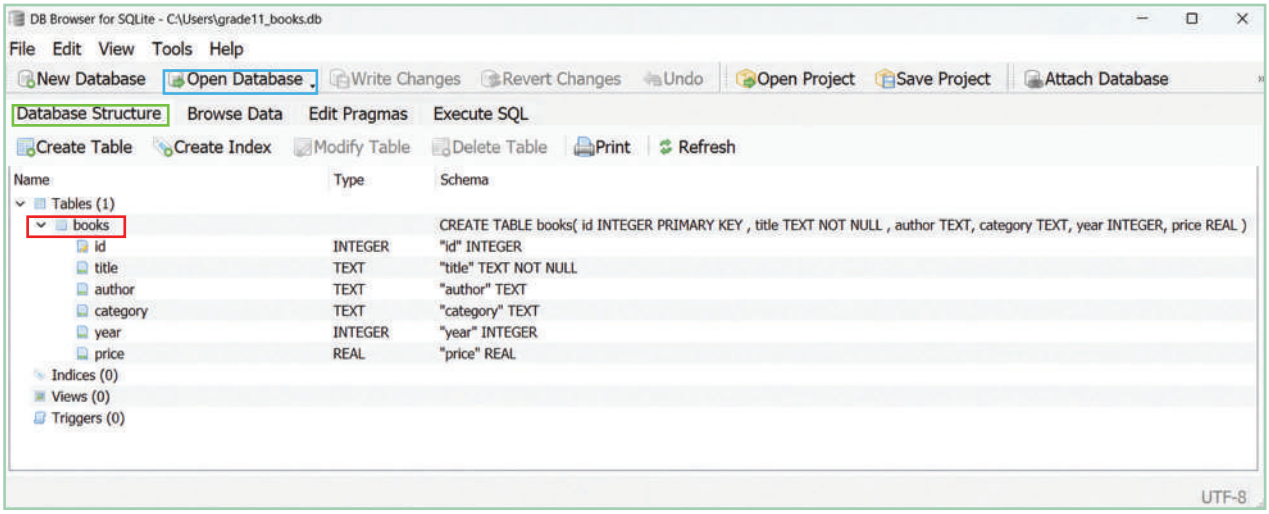
```
connection.close()
```

- 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 books من قائمة جداول قاعدة البيانات لتظهر تفاصيل هيكلية الجدول.

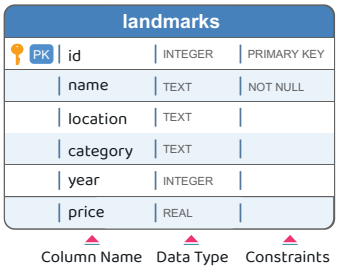


شكل رقم (3-4) يوضح خطوات معاينة هيكلية الجدول Table Structure.

أوراق العمل

ورقة عمل (3) التاريخ:

إنشاء جدول جديد landmarks بقاعدة البيانات Kuwait_landmarks.db لتخزين بيانات بعض معالم دولة الكويت.



شكل (3-5) يُمثل هيكلية جدول بقاعدة بيانات معالم الكويت.

تطبيق الخطوات التالية بالاسترشاد بالمخطط أعلاه:

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 connection.commit()
14 connection.close()
```

3. استكمل التعليمة البرمجية لإنشاء جدول landmarks، 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 (3-3) يُوضح جدول landmarks بقاعدة البيانات Kuwait_landmarks.db.

4. نفذ التعليمات البرمجية.

5. عاين هيكلية الجدول باستخدام برنامج DB Browser for SQLite.

قواعد البيانات SQLite في Python
إضافة البيانات Inserting Data

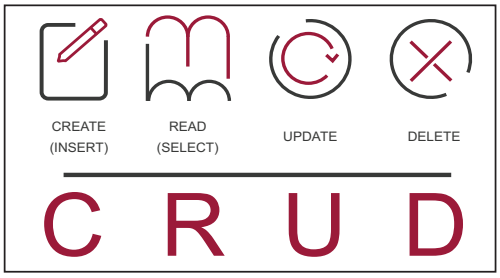
نواتج التعـلم

- التعرف على مفهوم عمليات CRUD، وكيفية تطبيقها على قواعد البيانات.
- إدراج صف واحد في قاعدة البيانات باستخدام الدالة ()execute.
- إدراج عدة صفوف دفعة واحدة باستخدام الدالة ()executemany.
- تطوير تعليمات برمجية للتعامل مع البيانات، 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:
   - استيراد وحدة 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 for SQLite.

إدخال صف واحد إلى الجدول

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

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

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

خطوات التنفيذ

- استيراد وحدة 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 for SQLite.

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

التفسير

```
import sqlite3
```

- استيراد وحدة sqlite3.

```
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 books(title,author,category,year,price)
        VALUES( ?,?,?,?,?)''',
        ('Arabic','Arabic SD','Art/Science',2001,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 كعنصر مؤقت من لغة Python إلى قاعدة بيانات SQLite، يتم تحويلها تلقائيًا إلى NULL داخل قاعدة البيانات، والتي تُستخدم لتمثيل غياب القيمة أو عدم توفر بيانات مُحددة للحقل.
  - لا ينصح بإدخال البيانات مباشرة (دون استخدام العنصر المؤقت) كالتالي:

```
cursor.execute('''
        INSERT INTO books(title, author,year,category,price)
        VALUES('Arabic','Arabic SD', 2001, 'Art/Science',NULL)
        ''')
```

يتم استخدام NULL بدل None عند إدخال القيم مباشرة.

```
connection.commit()
```

- حفظ التغييرات في قاعدة البيانات.

```
connection.close()
```

- إغلاق الاتصال بقاعدة البيانات.

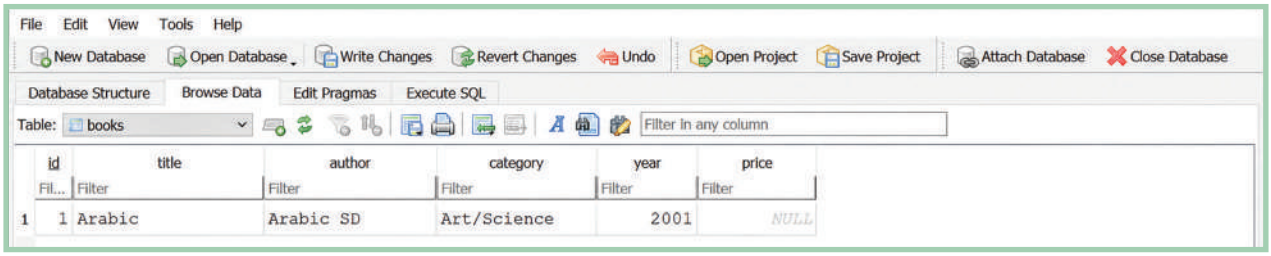
ملاحظات:

- تتولى قاعدة البيانات إسناد قيم رقمية تلقائية تصاعدية للمفتاح الأساسي من النوع INTEGER عند عدم إدخال قيم يدوية.
- عند تنفيذ الأمر INSERT دون إدخال بيانات لبعض الأعمدة، يتم تعبئة هذه الأعمدة بالقيمة الافتراضية DEFAULT إن وُجدت، وإلا تُستخدم NULL تلقائيًا، ما لم يكن العمود يحتوي على القيد NOT NULL أو INTEGER PRIMARY KEY.
- 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 for SQLite للتأكد من إضافة الصف إلى الجدول.



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

إدخال صفوف 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 books بقاعدة البيانات grade11_books.db.

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

جدول رقم (4-2) يوضح بيانات إدخال صفوف متعددة إلى جدول books 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 books table
6   cursor.executemany('''
7           INSERT INTO books(title, author,category,year,price)
8           VALUES( ?,?,?,?,?)''',
9       [
10          ('ICT', 'ICT SD','Art/Science',2001,None),
11          ('Physics', 'Science SD', 'Science',2007,1.500),
12          ('English', 'English SD','Art/Science',2009,None)
13      ]
14  )
15  # Commit changes
16  connection.commit()
17  # Close database
18  connection.close()
```

التفسير

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

- استخدم الدالة cursor.executemany : لإدراج عدة صفوف rows في الجدول دفعة واحدة.
- استخدام كائن قابل للتكرار Iterable مثل القائمة، والتي تحتوي على عدة عناصر من النوع Tuple :

◄ عدد العناصر يساوي عدد الصفوف المطلوب إضافتها.

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

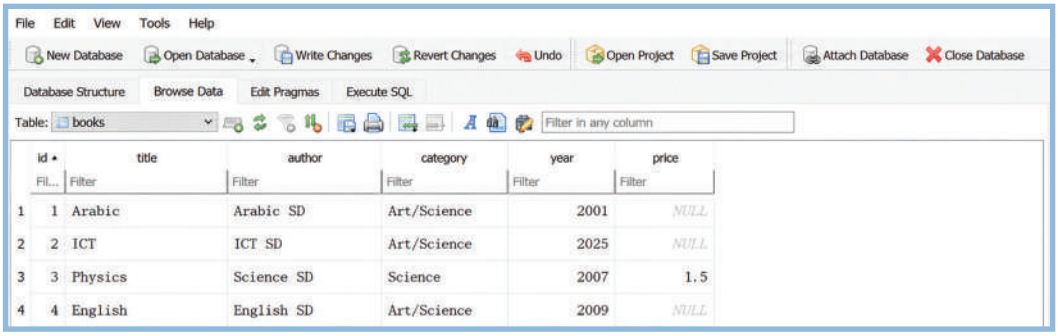
ملاحظات:

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

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

استخدام برنامج DB Browser for SQLite

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



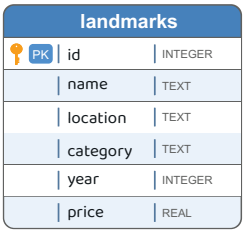
	id	title	author	category	year	price
1	1	Arabic	Arabic SD	Art/Science	2001	NULL
2	2	ICT	ICT SD	Art/Science	2025	NULL
3	3	Physics	Science SD	Science	2007	1.5
4	4	English	English SD	Art/Science	2009	NULL

شكل رقم (3-4) يوضح استخدام برنامج DB Browser for SQLite لمعاينة بيانات الصفوف في جدول books.

أوراق العمل

ورقة عمل (4) التاريخ:

إضافة صف لجدول landmarks بقاعدة البيانات Kuwait_landmarks.db.



شكل (4-4) يُمثل هيكلية جدول بقاعدة بيانات معالم الكويت.

تطبيق الخطوات التالية بالاسترشاد بالمخطط أعلاه:

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 landmarks table
6   cursor.execute(
7
8
9   )
10  # Commit changes
11  connection.commit()
12  # Close database
13  connection.close()
```

3. استكمل التعليمة البرمجية لإضافة صف وفقًا للبيانات التالية:

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

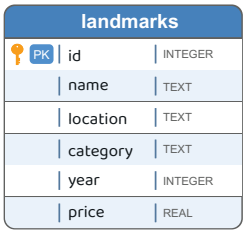
جدول رقم (4-3) يوضح بيانات صف مطلوب إضافته لجدول landmarks بقاعدة البيانات Kuwait_landmarks.db.

4. نفذ التعليمات البرمجية.

5. عاين بيانات الجدول باستخدام برنامج DB Browser for SQLite.

ورقة عمل (5) التاريخ:

إضافة عدة صفوف مرة واحدة لجـدول landmarks بقاعدة البيانـــات Kuwait_landmarks.db.



شكل (5-4) يُمثل هيكلية جدول بقاعدة بيانات معالم الكويت.

تطبيق الخطوات التالية بالاسترشاد بالمخطط أعلاه:

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

```
1  import sqlite3
2  connection = None
3  try:
4      connection=sqlite3.connect('Kuwait_landmarks.db')
5      cursor=connection.cursor()
6      # Insert data into landmarks table
7      insert_sql='.....'
8
9      landmarks_list = [('Failaka Island', 'Off the coast', 'Historical', 1950,5),
10             ('Souq Al-Mubarakiya','Central Kuwait City','Traditional',1897,0),
11             ('Museum of Modern Art','Salmiya Area','Art',1980,1)]
12     cursor.executemany(..... , .....)
13     connection.commit()
14 except Exception as e :
15     print('Error :', e)
16 finally :
17     if connection is not None:
18         connection.close()
19         print('Database connection is closed.')
20     else:
21         print('No database connection was created.')
```

3. استكمل التعليمة البرمجية لإضافة الصفوف المخصصة للمتغير landmarks_list.

Column names	id	name	location	category	year	price
Data values		Failaka Island	Off the coast	Historical	1950	5
		Souq Al-Mubarakiya	Central Kuwait City	Traditional	1897	0
		Museum of Modern Art	Salmiya Area	Art	1980	1

جدول رقم (4-4) يوضح بيانات صفوف مطلوب إضافتها لجدول landmarks بقاعدة البيانات Kuwait_landmarks.db.

4. نفذ التعليمات البرمجية.

5. عاين بيانات الجدول باستخدام برنامج DB Browser for SQLite.

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 وحدة 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 من القيم

العوامل الشرطية:

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

مثال:

price IN (10,20,25)

النوع: خارج قائمة من القيم

العوامل الشرطية:

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

مثال:

price NOT IN (10,20,25)

النوع: التحقق من غياب القيم

العوامل الشرطية:

IS NULL

مثال:

title IS NULL

النوع: التحقق من وجود القيم

العوامل الشرطية:

IS NOT NULL

مثال: title IS NOT NULL

title IS NOT NULL AND price IS NOT NULL

النوع: يُطابق نمط جزئي لنص

العوامل الشرطية:

LIKE 'pattern' ⟶

% : تُمثل مجموعة أحرف مجهولة _ : تُمثل حرف مجهول

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

النوع: لا يُطابق نمط جزئي لنص

العوامل الشرطية:

NOT LIKE 'pattern' ⟶

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

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

ملاحظات:

- توجد فروقات في صياغة العبارات الشرطية بين لغة Python وقواعد البيانات SQLite فكلٌ منهما يستخدم طريقة خاصة به لكتابة الشروط.
- غياب القيمة NULL تعني أن الحقل لا يحتوي على أي قيمة (غير معروفة أو غير مُدخلة) وهي ليست عددية صفر (0)، أو نصية فارغة (' ').

صيغة التعليمة البرمجية لإنشاء استعلام SQLite في Python

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

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.

- استيراد وحدة 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 books بقاعدة البيانات grade11_books.

خطوات التنفيذ

- استيراد وحدة 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 books')   #(*) means all fields
7   rows=cursor.fetchall()                  # Fetches all rows(books) as a list of tuples
8   # print rows in separate lines
9   for row in rows:
10      print(row)
11  #close database
12  connection.close()
```

التفسير

```
cursor.execute('SELECT * FROM books')
```

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

```
rows = cursor.fetchall()
```

- تسترجع جميع النتائج الناتجة عن الاستعلام وتخزنها في متغير rows في الذاكرة.
- نوع المُتغير rows: قائمة List تحتوي على صفوف Tuples ، ويُمثل كل Tuple صفًا من الجدول.

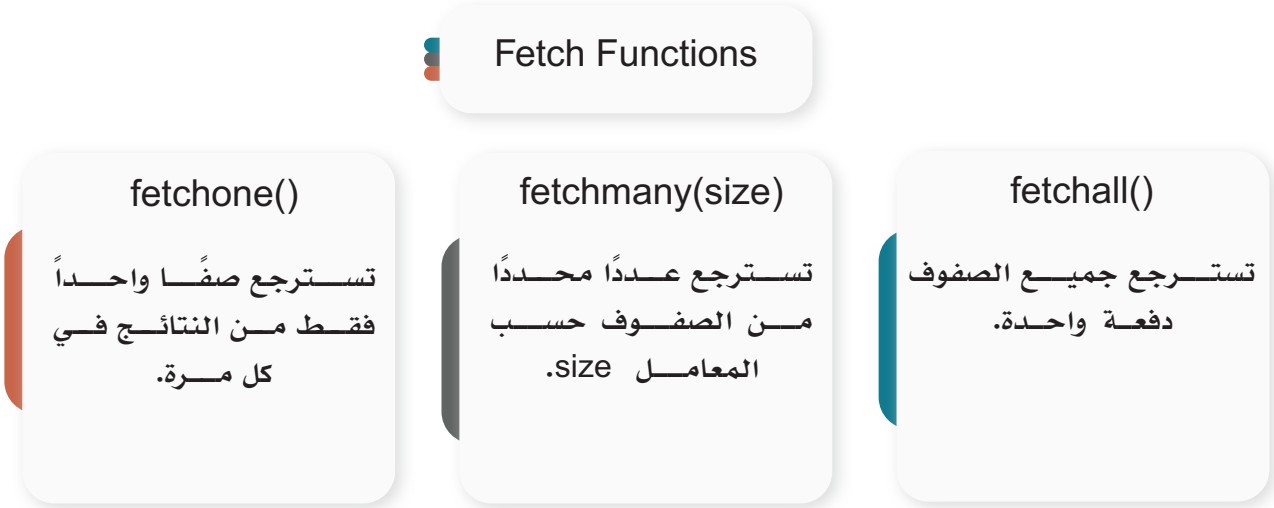
```
for row in rows:
    print(row)
```

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

ناتج تنفيذ التعليمة البرمجية (البيانات التي تم استرجاعها)

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

طرق استخدام دوال fetch



شكل رقم (3-5) 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.fetchall()
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 books بقاعدة البيانات grade11_books.db.

خطوات التنفيذ

- استيراد وحدة 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  books')
7   rows=cursor.fetchall()
8   for row in rows:
9       print(row)
10  #close database
11  connection.close()
```

التفسير

```
cursor.execute('SELECT title, price FROM books')
```

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

```
rows = cursor.fetchall()
```

- استرجاع جميع النتائج الناتجة عن الاستعلام.
- يتم تخزين النتائج في متغير rows على شكل قائمة List عناصرها Tuples، كل Tuple يُمثل صفًا (أي كتابًا) يحتوي على قيمتي العنوان والسعر فقط.

```
for row in rows:
    print(row)
```

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

Program Output

```
('Arabic', 1.5)
('French', 1.5)
('English', 1.5)
('Mathematics', 1.5)
('Mathematics', 1.5)
('Principles of Geography and Economics', 1.5)
('Psychology and sociology', 1.5)
('Islamic History', 1.5)
('Chemistry', 1.5)
('Physics', 0.0)
('Biology', 1.5)
('Geology', 1.5)
('ICT', 1.5)
('Quran', 1.5)
('Islamic Studies', 1.5)
```

الاستعلام عن بيانات بناءً على شرط محدد

مثال 3: تحديد بيانات الصفوف (الكتب) وفق الشرط (طُبعت بعد عام 2013)، من جدول books بقاعدة البيانات grade11_books.db.

خطوات التنفيذ

- استيراد وحدة 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 books WHERE year>?',(2013,))
7   rows=cursor.fetchall()
8   for row in rows:
9       print(row)
10  #Close database
11  connection.close()
```

التفسير

```
cursor.execute('SELECT * FROM books WHERE year > ?',(2013,))
```

- تُستخدم SELECT * لاسترجاع جميع الأعمدة وفقًا لشرط الاستعلام.
- تُستخدم عبارة WHERE year > 2013 لتحديد شرط الاستعلام بحيث يتم استرجاع الصفوف الخاصة بالكتب التي تم طباعتها بعد عام 2013 فقط. (تم استبدال ? بالقيمة 2013).

```
rows = cursor.fetchall()
```

- تُسترجع جميع الصفوف المطابقة للشرط السابق وتُخزن في المتغير rows.
- النتيجة تكون عبارة عن قائمة List تحتوي على عدة عناصر من النوع Tuple، بحيث يُمثل كل Tuple صفًا كاملاً من بيانات كتاب في الجدول.

```
for row in rows:
    print(row)
```

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

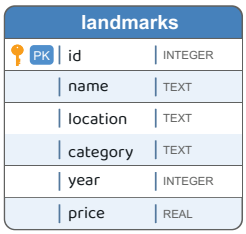
Program Output

```
(6, 'Principles of Geography and Economics', 'Social SD', 'Art', 2016, 1.5)
(7, 'Psychology and sociology', 'Social SD', 'Art', 2016, 1.5)
(13, 'ICT', 'ICT SD', 'Art/Science', 2025, 1.5)
```

أوراق العمل

ورقة عمل (6) التاريخ:

الاستعلام عن جميع البيانات من الجدول landmarks بقاعدة البيانات Kuwait_landmarks.db مستخدمًا التعليمة fetchall.



شكل (4-5) يُمثل هيكلية جدول بقاعدة بيانات معالم الكويت.

تطبيق الخطوات التالية بالاسترشاد بالمخطط أعلاه:

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()
```

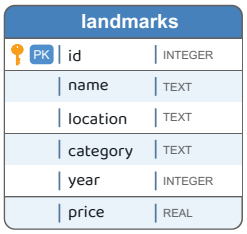
3. استكمل التعليمات البرمجية لتنفيذ التالي:
   - استعلم عن جميع صفوف الجدول.
   - اطبع الصفوف التي تم استرجاعها (الاستعلام عنها).

4. نفذ التعليمات البرمجية.

5. عاين نتائج الاستعلام.
   - عدد الصفوف: .....

ورقة عمل (7) التاريخ:

الاستعلام عن بيانات محددة من الجدول landmarks بقاعدة البيانات Kuwait_landmarks.db مُستخدمًا التعليمة fetchall.



شكل (5-5) يُمثل هيكلية جدول بقاعدة بيانات معالم الكويت.

تطبيق الخطوات التالية بالاسترشاد بالمخطط أعلاه:

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 rows.
   - اكتب التعليمة البرمجية للحلقة التكرارية للمتغير rows 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

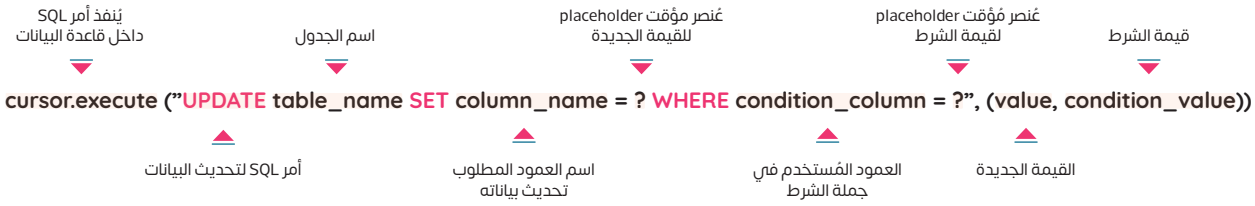
- شرح مفهوم تعديل البيانات في جداول قواعد البيانات باستخدام الأمر UPDATE.
- التمييز بين تحديث عمود أو أكثر في صف واحد أو أكثر.
- كتابة تعليمات برمجية لتحديث البيانات باستخدام وحدة sqlite3 في Python.
- تطبيق شروط بسيطة ومركبة لتحديث بيانات محددة.
- استخدام الدالة connection.total_changes لمعرفة عدد الصفوف 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 البيانات بقاعدة 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:

- استيراد وحدة 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 for SQLite.

تحديث بيانات عمود واحد بناءً على شرط بسيط.

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

خطوات التنفيذ

- استيراد وحدة 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 for SQLite.

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

التفسير

```
import sqlite3
```

- استيراد وحدة sqlite3 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 books SET price = ? WHERE id = ?',(2,5))
```

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

```
print('Number of rows updated = ', connection.total_changes)
```

- connection.total_changes معرفة عدد الصفوف التي تم تحديثها (تم تحديث صف واحد).

```
connection.commit()
```

- حفظ التغييرات في قاعدة البيانات.

```
connection.close()
```

- إغلاق الاتصال بقاعدة البيانات.

ملاحظة:

- استخدام الخاصية connection.total_changes لمعرفة عدد الصفوف التي تأثرت بعمليات الإضافة، التعديل والحذف التي تمت على قاعدة البيانات منذ بدء الاتصال.

تحديث بيانات أكثر من عمود بالجدول بناء على شرط بسيط.

مثال 2: تعديل بيانات أعمدة جدول books بقاعدة البيانات grade11_books.db.
ليصبح price = 2.5 و year = 2024 وفقًا للشرط category = 'Art'.

خطوات التنفيذ

- استيراد وحدة 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 for SQLite.

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

التفسير

```
import sqlite3
```

- استيراد وحدة sqlite3 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 books 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 = ', connection.total_changes)
```

- connection.total_changes معرفة عدد الصفوف التي تم تحديثها(تم تحديث 5 صفوف).

```
connection.commit()
```

- حفظ التغييرات في قاعدة البيانات.

```
connection.close()
```

- إغلاق الاتصال بقاعدة البيانات.

في المثال السابق:

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

```
cursor.execute('''
    UPDATE books
    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 books
    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:

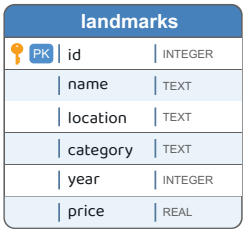
category = 'Art' و price < 2.5.

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

أوراق العمل

ورقة عمل (8) التاريخ:

تعديل / تحديث بيانات الجدول landmarks بقاعدة البيانات Kuwait_landmarks.db.



شكل (2-6) يُمثل هيكلية جدول بقاعدة بيانات معالم الكويت.

تطبيق الخطوات التالية بالاسترشاد بالمخطط أعلاه:

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:', connection.total_changes)
11  connection.commit()
12  connection.close()
```

3. استكمل التعليمات البرمجية لتنفيذ التالي:

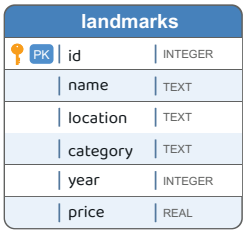
- حدث بيانات الصف id=1 لتصبح قيمة تذكرة الدخول تساوي دينارين price = 2.

4. نفذ التعليمات البرمجية.

- عدد الصفوف التي تم تعديلها = ...

ورقة عمل (9) التاريخ:

تعديل / تحديث بيانات الجدول landmarks بقاعدة البيانات Kuwait_landmarks.db.



شكل (3-6) يُمثل هيكلية جدول بقاعدة بيانات معالم الكويت.

تطبيق الخطوات التالية بالاسترشاد بالمخطط أعلاه:

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:', connection.total_changes)
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 Deleting Data، وأهميتها في قواعد البيانات.
- تنفيذ عملية حذف صف أو عدة صفوف محددة من جدول في قاعدة البيانات باستخدام أمر DELETE مع شرط Condition.
- 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.

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

حذف البيانات Deleting Data

تُعتبر عملية حذف البيانات من العمليات الأساسية في إدارة قواعد البيانات، حيث تهدف إلى إزالة صف 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 من قاعدة 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:
   - استيراد وحدة 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.
4. معاينة النتائج في برنامج DB Browser for SQLite.

حذف الصفوف بناء على شرط بسيط.

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

خطوات التنفيذ

- استيراد وحدة 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 for SQLite.

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

التفسير

```
import sqlite3
```

- استيراد وحدة sqlite3 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 books WHERE title=? ',('Mathematics',))
```

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

```
print('Number of rows deleted = ', connection.total_changes)
```

- connection.total_changes معرفة عدد الصفوف التي تم حذفها (تم حذف صفين).

```
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 books بقاعدة البيانات grade11_books.db، وفقًا للشرط price > 1 ، و year < 2013.

خطوات التنفيذ

- استيراد وحدة 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 for SQLite.

```
1  import sqlite3
2  connection = sqlite3.connect('grade11_books.db')
3  cursor = connection.cursor()
4  # Delete data
5  cursor.execute('DELETE FROM books WHERE price > ? AND year < ?',(1,2013))
6  print('Number of rows deleted =' , connection .total_changes)
7  connection.commit()
8  connection.close()
```

التفسير

```
import sqlite3
```

استيراد وحدة sqlite3 للتعامل مع قواعد البيانات.

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

إنشاء اتصال بقاعدة البيانات grade11_books.db.

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

إنشاء كائن المؤشر cursor لتنفيذ أوامر SQL داخل قاعدة البيانات.

```
cursor.execute('DELETE FROM books WHERE price > ? AND year < ?',(1,2013))
```

حذف الصفوف من جدول books إذا كان السعر أكبر من 1، وسنة النشر أقل من 2013.

```
print('Number of rows deleted = ', connection.total_changes)
```

connection.total_changes معرفة عدد الصفوف التي تم حذفها (تم 4 صفوف).

```
connection.commit()
```

حفظ التغييرات في قاعدة البيانات.

```
connection.close()
```

إغلاق الاتصال بقاعدة البيانات.

ملاحظة:

- لحذف الصفوف التي لا تحتوي على قيم في عمود تاريخ النشر (year).

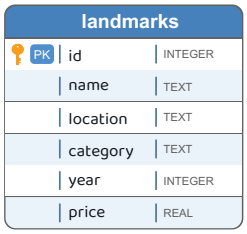
```
cursor.execute('DELETE FROM books WHERE year IS NULL')
```

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

أوراق العمل

ورقة عمل (10) التاريخ:

حذف صفـوف من الجدول landmarks بقاعدة البيانات Kuwait_landmarks.db.



شكل (2-7) يُمثل هيكلية جدول بقاعدة بيانات معالم الكويت.

تطبيق الخطوات التالية بالاسترشاد بالمخطط أعلاه:

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:', connection .total_changes)
12 # Commit changes
13 connection.commit()
14 # Close database
15 connection.close()
```

3. استكمل التعليمات البرمجية:

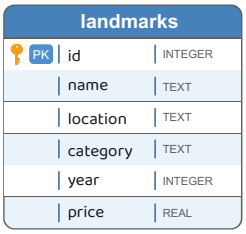
- احذف جميع صفوف معالم الكويت الثقافية. 'category = 'Cultural.

4. نفذ التعليمات البرمجية.

- عدد الصفوف التي تم حذفها = ...

ورقة عمل (11) التاريخ:

حذف صفـوف من الجدول landmarks بقاعدة البيانات Kuwait_landmarks.db.



شكل (7-3) يُمثل هيكلية جدول بقاعدة بيانات معالم الكويت.

تطبيق الخطوات التالية بالاسترشاد بالمخطط أعلاه:

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:', connection .total_changes)
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.

الوحدات والحزم المستخدمة في المشروع

الوحدة / الحزمة	الاستخدام	تثبيت الوحدة
OpenCV	لاكتشاف الوجوه والتعامل مع الصور	pip install opencv-python
os	لإدارة الملفات ومسارات الصور	وحدة 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

حزمة OpenCV

Open-Source Computer Vision Library

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

مراحل المشروع

إعداد بيئة المشروع:

- تجهيز وحدات وحزم Python مثل cv2 و sqlite3.
- إعداد مجلد يحتوي على صور الأشخاص المعروفين.
- التعرف على الوجه.
- استخدام حزمة 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_logs (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    image_name TEXT,
    person_name TEXT,
    x INTEGER,
    y INTEGER,
    w INTEGER,
    h INTEGER,
    detected_at TEXT
)
```

◄ الجدول face_logs 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 cv2.
- 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), (0, 255, 0), 2)
cv2.putText(image, name, (x, y + h + 25), ...)
```

7 حفظ النتائج.

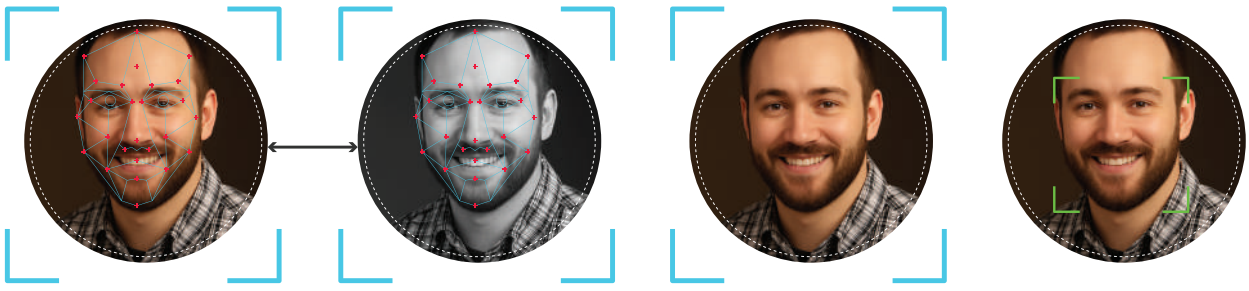
```
cursor.execute('''
    INSERT INTO face_logs (...)
    VALUES (?, ?, ?, ?, ?, ?, ?)
''', (...))
connection.commit()
connection.close()
```

◄ 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.



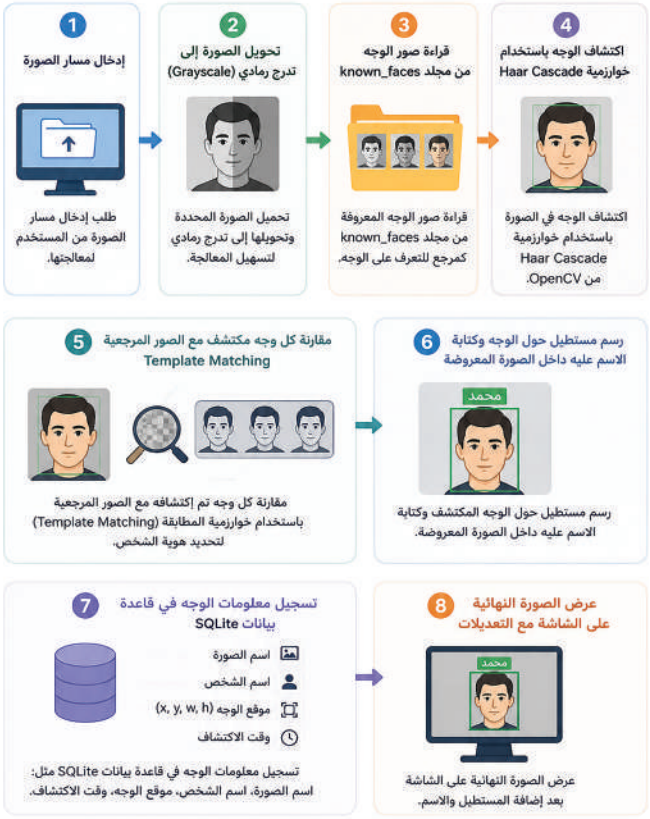
التعليمات البرمجية المقترحة لتنفيذ المشروع

```
# Import required libraries for image processing, file handling, database, and timestamps
import cv2
import os
import sqlite3
from datetime import datetime
# Function to recognize a face using template matching against stored images
def recognize_face_by_template(face_gray):
  best_match_name = 'Unknown'
  highest_score = 0
  # Loop through all images inside the known_faces directory
  for filename in os.listdir('known_faces'):
    print(f'Comparing with image: {filename}')
    # Load each known face image in grayscale
    known_image = cv2.imread(f'known_faces/{filename}', 0)
    # Skip if image failed to load
    if known_image is None:
      print(f'Failed to load image: {filename}')
      continue
    # Resize detected face to match known image size
    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
    # Apply template matching to compute similarity score
    result = cv2.matchTemplate(known_image, face_resized, cv2.TM_CCOEFF_NORMED)
    # Extract the highest similarity score
    _, score, _, _ = cv2.minMaxLoc(result)
    print(f'Match score with {filename}: {score:.2f}')
    # Update best match if score exceeds threshold
    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 SQLite database and prepare cursor
connection = sqlite3.connect('face_detections.db')
cursor = connection.cursor()
# Create table to store face detection logs if it does not exist
cursor.execute("""
  CREATE TABLE IF NOT EXISTS face_logs (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    image_name TEXT,
    person_name TEXT,
    x INTEGER,
    y INTEGER,
    w INTEGER,
```

```
    h INTEGER,
    detected_at TEXT
  )
''')
# Request image path from user and load the image
image_path = input('Enter image path (e.g., test.jpg or C:/Users/...): ').strip()
image = cv2.imread(image_path)
# Validate that image was loaded successfully
if image is None:
  print('Failed to load image. Please check the path.')
  exit()
# Convert image to grayscale for face detection
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
# Load Haar cascade model and detect faces in the image
face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_frontalface_default.xml')
faces = face_cascade.detectMultiScale(gray, scaleFactor=1.1, minNeighbors=5)
print(f'Number of detected faces: {len(faces)}')
# Process each detected face: crop, recognize, draw, and store in database
for (x, y, w, h) in faces:
  x = int(x)
  y = int(y)
  w = int(w)
  h = int(h)
  # Extract face region from grayscale image
  face_crop = gray[y:y+h, x:x+w]
  # Recognize the face using template matching function
  name = recognize_face_by_template(face_crop)
  print(f'Detected face at ({x},{y}) recognized as: {name}')
  # Draw rectangle around detected face
  cv2.rectangle(image, (x, y), (x + w, y + h), (0, 255, 0), 2)
  # Put recognized name below the face
  cv2.putText(image, name, (x, y + h + 25), cv2.FONT_HERSHEY_SIMPLEX, 0.8, (0, 255, 0), 2)
  # Generate timestamp for logging
  timestamp = datetime.now().strftime('%Y-%m-%d %H:%M:%S')
  # Insert detection record into database
  cursor.execute('''
    INSERT INTO face_logs (image_name, person_name, x, y, w, h, detected_at)
    VALUES (?, ?, ?, ?, ?, ?, ?)
  ''', (os.path.basename(image_path), name, x, y, w, h, timestamp))
# Save changes and close database connection
connection.commit()
connection.close()
print('Face data saved to database.')
# Display the final image with detected faces
cv2.imshow('Identified Faces', image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

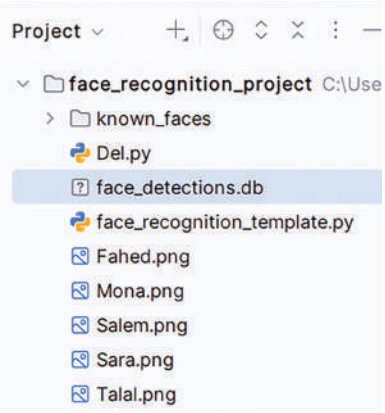
الإجراءات التي تتم عند تشغيل البرنامج

1. طلب إدخال مسار الصورة من المستخدم لمعالجتها.
2. تحميل الصورة المحددة وتحويلها إلى تدرج رمادي Grayscale لتسهيل المعالجة.
3. قراءة صور الوجوه المعروفة من مجلد known_faces كمرجع للتعرف على الوجه.
4. اكتشاف الوجوه في الصورة باستخدام خوارزمية Haar Cascade من OpenCV.
5. مقارنة كل وجه مُكتشف مع الصور المرجعية باستخدام template matching لتحديد هوية الشخص.
6. رسم مستطيل حول الوجه وكتابة الاسم عليه داخل الصورة المعروضة.
7. تسجيل معلومات الوجه في قاعدة بيانات SQLite ، مثل:
   - اسم الصورة.
   - اسم الشخص.
   - موقع الوجه (x, y, w, h).
   - وقت الكشف عن الوجه.
8. 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.

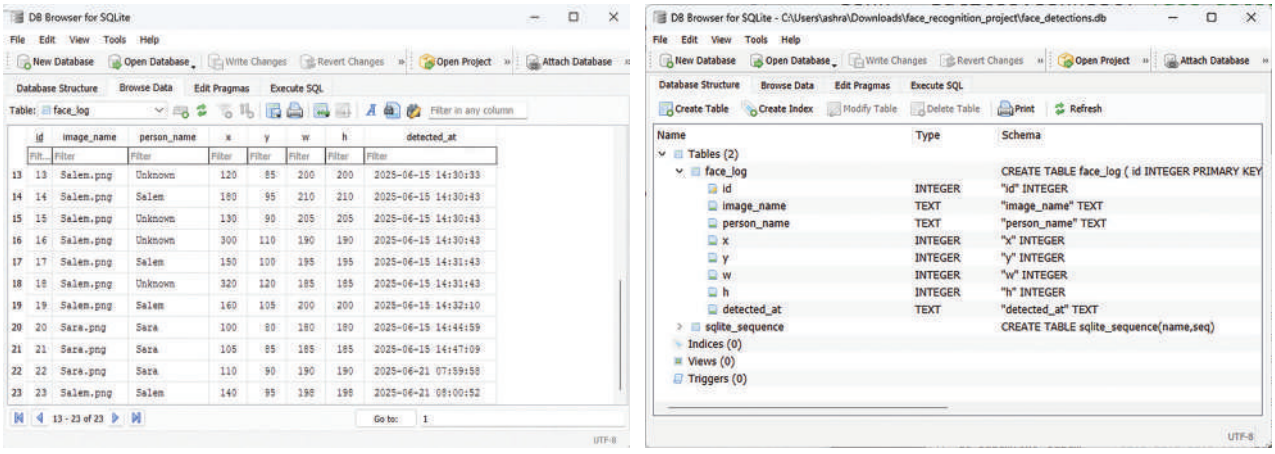


شكل رقم (8-1) يُوضح مكان إنشاء قاعدة البيانات.

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 for SQLite.

التقنيات المُستخدمة في المشروع

1 الفرق بين تقنية كشف الوجه وتقنية التعرف عليه.

كشف الوجه Face Detection:
عمليــة تحديــد موقــع وجــود وجــه داخــل صــورة أو فيديــو، أي تحديــد الإحداثيــات (x, y)، والأبعــاد الطــول والعــرض (h, w).

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

2 استخدامات وحدة CV2 وحزمة OpenCV 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.

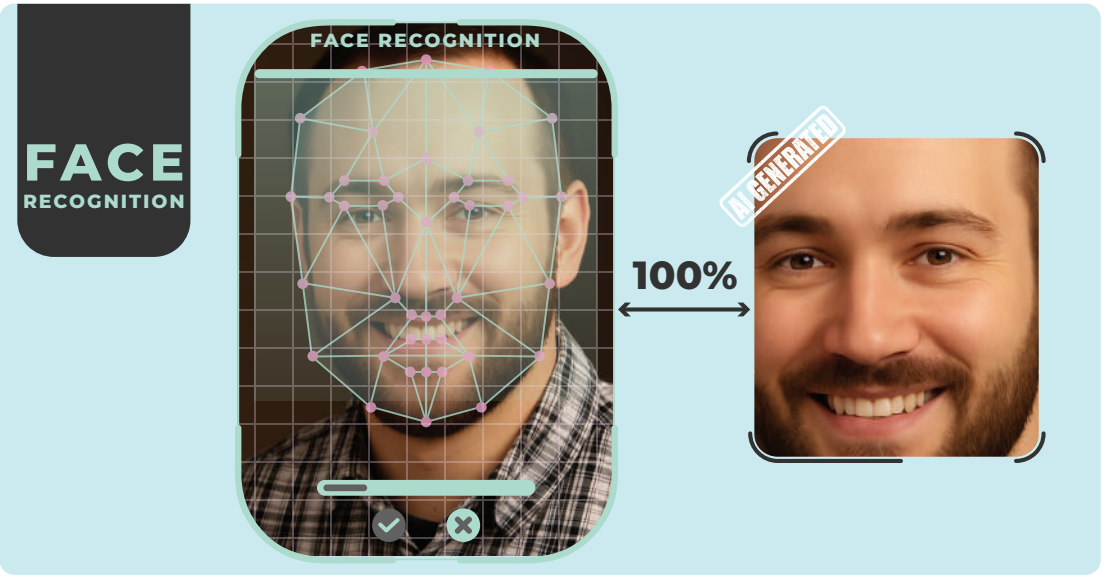
- يتم تحويل الصورة إلى تدرج رمادي لأن خوارزميات كشف الوجوه والتعرف عليها تعمل بشكل أسرع وأكفأ على الصور الرمادية (ذات القناة الواحدة) بدلاً من الصور الملونة (ذات 3 قنوات).
- تقليل حجم البيانات المراد معالجتها.
- معظم الكواشف والتقنيات تعتمد على تباين الإضاءة وليس الألوان.

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 SQLite 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 cv2 تقوم بمقارنة صورة صغيرة (الوجه المُكتشف) مع صورة كبيرة (الوجه المعروف).
- تعيد الدالة قيمة عددية (بين 0 و 1) تمثل درجة التشابه بين الصورتين، حيث تشير القيم الأقرب إلى 1 إلى تطابق أعلى.



المنتجات الرقمية

مراحل تصميم وتطوير المنتج الرقمي

Designing and Developing a Digital Product

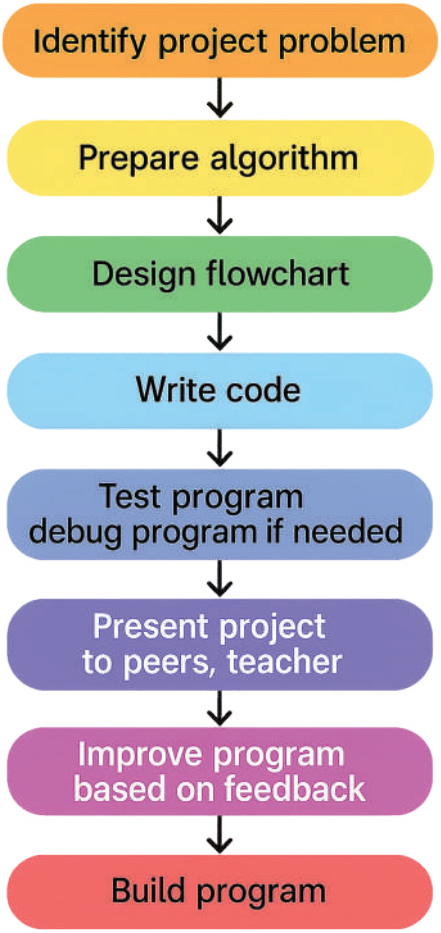
نواتج التعـلم

- 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 لإنتاج 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 قاعدة البيانات:** المسؤول عن إعداد قاعدة البيانات.
- **المُبرمج:** المسؤول عن كتابة التعليمات البرمجية 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 تصميم قاعدة البيانات.
		☐ 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 تقنيات الذكاء الاصطناعي.

سجل فكرة المشروع بعد مناقشة أعضاء الفريق:

أهداف المشروع

يهدف المشروع إلى إكساب المتعلم مهارات تقنية وإبداعية عبر تصميم وتنفيذ فكرة تعليمية بطريقة منظمة 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 الوحدات والحزم اللازمة.
- PowerPoint: لإعداد العرض التقديمي.
- Windows screen recorder: لتسجيل الشاشة إستعداداً لتوثيق المشروع.
- Draw.io: لرسم خريطة التدفق للمشروع.
- DB Broswer for SQLite: التعامل مع قاعدة البيانات.
- Microsoft Edge: للبحث عن المعلومات.

وأي 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.
- توظيف مهارات قواعد البيانات المختلفة.

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 كيف تم توظيف الذكاء الاصطناعي في المشروع، مع ذكر اسم النموذج المُستخدم.
- 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

الكتاب كاملاً

قيِّم مناهجنا