https://datascience-project.eu/

Ausbildung Fiche

Titel	Software
Schlüsselwörter (meta tag)	SQL Github
Sprache	Deutsch
Zielsetzungen / Ziele / Lernergebnisse	1. **Verstehen, wann man SQL verwenden sollte** 2. **Wichtige Anwendungsbereiche von SQL** 3. **Die grundlegenden SQL-Befehle** 4. **Wie man gemeinsam im Team Code entwickelt** 5. **Grundlegende Funktionalitäten von GitHub**
Lehrgang:	
Datenwissenschaftliche Kompetenz	
Modul Datenvisualisierung und visuelle Analyse	
Einführung in die Datenwissenschaft für Human- und Sozialwissenschaften	
Datenwissenschaft für den guten Zweck	
Datenjournalismus und Geschichtenerzählen	
• **Beschreibung**	• Dieser Kurs gibt eine kurze Einführung in die wichtigsten Programmiersprachen und Tools, die Data Scientists täglich verwenden. • Es wird erläutert, in welchem Zusammenhang und zu welchem Zweck sie üblicherweise verwendet werden, und es werden die wichtigsten Befehle für Anfänger:innen vorgestellt: ○ **SQL** ist zu einem Eckpfeiler der modernen Datenverwaltung geworden. In diesem Kurs werden wir verschiedene Möglichkeiten erkunden, wie SQL zum Abrufen von Daten aus Datenbanken verwendet werden kann. ○ Wir werden besprechen, was **GitHub** ist, welche Funktionen es bietet und wie Softwareentwickler davon profitieren können. • Am Ende des Kurses kennen die Teilnehmer:innen das Tätigkeitsfeld und die gebräuchlichsten Befehle.

Co-funded by the Erasmus+ Programme of the European Union

Mit Unterstützung des Programms Erasmus+ der Europäischen Union. Dieses Dokument und sein Inhalt spiegeln die
Die Verantwortung für den Inhalt dieser Veröffentlichung trägt allein der Verfasser; die Kommission haftet nicht für die weitere Verwendung der darin enthaltenen Angaben.

Inhalt in 3 Ebenen gegliedert	**2. Einführung in SQL** **2.1 Hintergrundinformationen** **2.1.1 SQL: die weltweite Standardsprache für die Verwaltung und Analyse von Datenbanken** Structured Query Language (SQL) ist eine weit verbreitete Datenbanksprache, die für die Verwaltung und Bearbeitung relationaler Datenbanken konzipiert ist. Sie ist ein leistungsfähiges Werkzeug, mit dem Benutzer:innen Daten auf verschiedene Weise analysieren, manipulieren und verarbeiten können, was sie zu einer beliebten Wahl in vielen Branchen und Anwendungsbereichen macht, darunter Finanzen, E-Commerce, Gesundheitswesen und Behörden. Eine der wichtigsten Eigenschaften von SQL ist seine Flexibilität. Es bietet eine Standardmethode für die Pflege und Verarbeitung großer Datenmengen und ist damit die erste Wahl für Unternehmen, die große Mengen an Informationen verwalten müssen. Dank seiner Flexibilität lassen sich auch individuelle Abfragen und Berichte erstellen, die den Nutzern wertvolle Einblicke in die Daten geben, mit denen sie arbeiten. Neben seiner Flexibilität zeichnet sich SQL auch durch seine Fähigkeit aus, Daten schnell und effizient zu verarbeiten und zu manipulieren. Durch den Einsatz komplexer SQL-Anweisungen und Indizierungstechniken kann SQL Daten schnell auffinden und abrufen, was besonders bei der Arbeit mit großen Datenbanken wichtig ist. Ein weiterer Vorteil von SQL ist seine Benutzerfreundlichkeit. Für die meisten Benutzer:innen ist es relativ einfach zu erlernen und zu verwenden, was es zu einer beliebten Wahl für Unternehmen und Organisationen jeder Größe macht. Außerdem gibt es zahlreiche Quellen, um die Sprache zu erlernen, von Online-Tutorials und -Kursen bis hin zu Lehrbüchern und Handbüchern. Trotz seiner vielen Vorteile hat SQL auch einige Einschränkungen. Zum Beispiel ist es nicht immer die beste Wahl für die Arbeit mit unstrukturierten Daten, wie Bildern, Videos oder Audiodateien. Außerdem ist es möglicherweise nicht die effektivste Wahl für bestimmte Arten von Analysen, z. B. solche, die fortgeschrittene statistische Techniken erfordern. **2.1.2 Zweck von SQL**

Co-funded by the Erasmus+ Programme of the European Union

Mit Unterstützung des Programms Erasmus+ der Europäischen Union. Dieses Dokument und sein Inhalt spiegeln die
Die Verantwortung für den Inhalt dieser Veröffentlichung trägt allein der Verfasser; die Kommission haftet nicht für die weitere Verwendung der darin enthaltenen Angaben.

Eine der Hauptanwendungen von SQL ist der Abruf von Daten aus einer Datenbank. Dies kann die Auswahl bestimmter Datenspalten, das Filtern von Daten nach bestimmten Kriterien oder die Kombination von Daten aus mehreren Tabellen umfassen. Nehmen wir zum Beispiel an, wir haben eine Datenbank mit Kunden und deren Bestellungen. Mit SQL können wir leicht eine Liste aller Bestellungen für eine:n bestimmte:n Kund:in oder alle Bestellungen für einen bestimmten Zeitraum abrufen.

Eine weitere Anwendung von SQL ist das Hinzufügen neuer Daten zu einer Datenbank, das Bearbeiten vorhandener Daten oder das Löschen nicht mehr benötigter Daten. Dies kann besonders nützlich sein, wenn wir große Datenmengen auf einmal aktualisieren müssen. Wenn wir z. B. die Lieferadresse für alle Kund:innen, die in einer bestimmten Postleitzahl wohnen, aktualisieren müssen, können wir diese Änderung schnell und einfach mit SQL vornehmen.

Neben der Verwaltung von Daten kann SQL auch dazu verwendet werden, ganze Tabellen in Datenbanken zu erstellen und zu verwalten. Dazu gehört das Erstellen neuer Tabellen, das Ändern bestehender Tabellen und das Löschen von Tabellen, die nicht mehr benötigt werden. Wenn wir z. B. eine neue Tabelle zur Erfassung der Verkaufsdaten eines Unternehmens erstellen möchten, können wir mit SQL die Struktur der Tabelle definieren und die Datentypen für jede Spalte festlegen.

Schließlich ist SQL besonders gut für die Verarbeitung großer Datenmengen geeignet. Das liegt daran, dass es auf hohe Effizienz ausgelegt ist und komplexe Abfragen und Operationen mühelos bewältigen kann. Damit ist es die beste Wahl für Unternehmen und Organisationen, die regelmäßig große Datenmengen verwalten und analysieren müssen.

2.1.3 Es gibt nur 4 wesentliche Dinge, die wir benötigen, um SQL sinnvoll zu nutzen

Wenn du mit der Verwendung von SQL beginnen möchtest, fragst du dich vielleicht, welche Werkzeuge und Kenntnisse du für den Einstieg benötigst. Zum Glück brauchst du nur 4 wesentliche Dinge.

Zunächst benötigst du ein Datenbankmanagementsystem (DBMS). Ein DBMS ist ein Softwaresystem, mit dem du

Co-funded by the Erasmus+ Programme of the European Union

Mit Unterstützung des Programms Erasmus+ der Europäischen Union. Dieses Dokument und sein Inhalt spiegeln die
Die Verantwortung für den Inhalt dieser Veröffentlichung trägt allein der Verfasser; die Kommission haftet nicht für die weitere Verwendung der darin enthaltenen Angaben.

Datenbanken erstellen, verwalten und manipulieren kannst. Es gibt viele verschiedene DBMS, aber zu den beliebtesten gehören MySQL, Oracle, PostgreSQL und Microsoft SQL Server. Diese Systeme bieten die Möglichkeit, Daten zu organisieren und zu speichern sowie mit Hilfe von SQL auf diese Daten zuzugreifen und sie zu bearbeiten.

Das **zweite,** was du brauchst, ist eine Datenbank. Eine Datenbank ist eine Sammlung von Daten, die in einer bestimmten Weise organisiert sind, um den Zugriff und die Bearbeitung zu erleichtern. Es gibt viele verschiedene Arten von Datenbanken, aber zu den beliebtesten gehören Oracle, PostgreSQL, MySQL und SQL Server. Wir können die Open-Source-Datenbank PostgreSQL einfach herunterladen: https://www.postgresql.org/

Als **Drittes** benötigen wir einen SQL-Client. Ein SQL-Client ist ein Werkzeug, mit dem du eine Verbindung zu einer Datenbank herstellen und SQL-Anweisungen ausführen kannst. Es gibt viele verschiedene SQL-Clients, aber zu den beliebtesten gehören MySQL Workbench, SQL Developer und SQL Server Management Studio. Alternativ können wir auch eine Programmiersprache wie Java, Python oder C# verwenden, um SQL-Anweisungen für eine Datenbank auszuführen.

Viertens und letztens benötist du Grundkenntnisse der SQL-Syntax und -Konzepte. Dazu gehört, dass wir wissen, wie man Tabellen erstellt und manipuliert, wie man SELECT-, INSERT-, UPDATE- und DELETE-Anweisungen verwendet, um mit Daten zu interagieren, und wie man WHERE-Klauseln zum Filtern von Daten verwendet. Sobald wir diese Konzepte verstanden haben, können wir SQL verwenden, um Daten in einer Vielzahl von Situationen zu extrahieren, zu bearbeiten und zu verwalten.

2.2 Relationale Datenbanken sind der Schlüssel

Relationale Datenbanken sind ein grundlegender Bestandteil der modernen Datenverwaltung in Unternehmen jeder Größe. Sie werden zum Speichern, Verwalten und Analysieren von Daten in einer Vielzahl von Umgebungen verwendet, von kleinen Unternehmen bis hin zu großen Konzernen, Regierungsbehörden und gemeinnützigen Organisationen.

Einer der Hauptvorteile einer relationalen Datenbank besteht darin, dass große Datenmengen strukturiert und organisiert

Co-funded by the Erasmus+ Programme of the European Union

Mit Unterstützung des Programms Erasmus+ der Europäischen Union. Dieses Dokument und sein Inhalt spiegeln die
Die Verantwortung für den Inhalt dieser Veröffentlichung trägt allein der Verfasser; die Kommission haftet nicht für die weitere Verwendung der darin enthaltenen Angaben.

gespeichert werden, so dass die Daten sehr schnell abgerufen (gefunden) werden können. Du kannst zum Beispiel eine relationale Datenbank verwenden, um alle Daten der Mitarbeiter Ihres Unternehmens zu speichern und zu verwalten.

Relationale Datenbanken beruhen auf einer einfachen, aber wirkungsvollen Idee: Daten können in Tabellen gespeichert werden, die aus Zeilen und Spalten bestehen. Jede **Tabelle** steht für eine bestimmte Art von Daten, und eine **Zeile** in der Tabelle steht für ein einzelnes Element. In unserer Tabelle mit den Angestellten zum Beispiel enthält jede Zeile eine andere Person, und jede **Spalte** steht für ein anderes **Attribut** des Angestellten (z. B. Name, Adresse und Telefonnummer). Eine Zeile wird auch als **Datensatz bezeichnet**, und die einzelnen Attribute werden in **Feldern** gespeichert. Die Kopfzeile der Tabelle wird auch als **Schema bezeichnet**, da hier der Bezug besteht, welche Daten in welcher Spalte gespeichert sind.

Ein weiterer Vorteil einer relationalen Datenbank ist die Möglichkeit, die Integrität und Konsistenz der Daten zu gewährleisten. Das bedeutet, dass du Regeln und Einschränkungen für die Eingabe von Daten in die Datenbank festlegen und dann sicherstellen kannst, dass diese Regeln eingehalten werden. So kannst du beispielsweise verlangen, dass für alle Mitarbeiter:innen eine Adresse oder eine eindeutige Telefonnummer gespeichert wird. Dadurch werden Datenfehler und Inkonsistenzen vermieden, die zu Problemen führen können.

Es gibt noch andere Arten von Datenbanken. NoSQL-Datenbanken zum Beispiel sind für die Verarbeitung unstrukturierter und halbstrukturierter Daten wie Dokumente, Diagramme und Schlüssel-Wert-Paare konzipiert. Sie werden häufig für Big Data-Analysen und Echtzeit-Webanwendungen verwendet.

Andere Typen sind objektorientierte Datenbanken, Graph-Datenbanken oder In-Memory-Datenbanken. Die Wahl des Datenbanktyps hängt von den spezifischen Bedürfnissen und Anforderungen der jeweiligen Anwendung ab.

Co-funded by the Erasmus+ Programme of the European Union

Mit Unterstützung des Programms Erasmus+ der Europäischen Union. Dieses Dokument und sein Inhalt spiegeln die
Die Verantwortung für den Inhalt dieser Veröffentlichung trägt allein der Verfasser; die Kommission haftet nicht für die weitere Verwendung der darin enthaltenen Angaben.

2.3 SQL-Anweisungen

2.3.1 Struktur und Syntax von SQL-Anweisungen

In diesem Skript lernen wir die in SQL-Anweisungen verwendeten **Schlüsselwörter** (auch als Klausel bezeichnet) sowie die allgemeine **Struktur und Syntax** von SQL-Anweisungen kennen.

Schlüsselwörter geben die Art der auszuführenden Operation an. Im Folgenden sind die am häufigsten in SQL-Anweisungen verwendeten Schlüsselwörter aufgeführt:

SELECT: Dient zum Abrufen von Daten aus einer oder mehreren Tabellen
INSERT: Dient zum Einfügen neuer Daten in eine Tabelle
UPDATE: Dient zur Aktualisierung vorhandener Daten in einer Tabelle
DELETE: Dient zum Löschen von Daten aus einer Tabelle
ALTER: Wird verwendet, um die Struktur einer Tabelle zu ändern
DROP: Dient zum Löschen einer Tabelle oder eines Datenbankobjekts
CREATE: Zum Erstellen einer neuen Tabelle oder eines neuen Datenbankobjekts
USE: Zur Auswahl einer Datenbank, mit der gearbeitet werden soll
SHOW: Dient zur Anzeige von Informationen über ein Datenbankobjekt

Die **Struktur und Syntax** von SQL-Anweisungen kann je nach verwendetem Datenbankmanagementsystem (DBMS) variieren. Es gibt jedoch einige allgemeine Richtlinien, die für die meisten SQL-Anweisungen gelten. Eine grundlegende SQL-Anweisung besteht normalerweise aus den folgenden Elementen:

Schlüsselwort: Das Schlüsselwort oder die Klausel, die die Art des auszuführenden Vorgangs angibt

Argumente: Ein oder mehrere Argumente oder Parameter, die zusätzliche Informationen über den auszuführenden Vorgang liefern

Semikolon: Alle SQL-Anweisungen enden mit einem Semikolon (;)

Co-funded by the Erasmus+ Programme of the European Union

Mit Unterstützung des Programms Erasmus+ der Europäischen Union. Dieses Dokument und sein Inhalt spiegeln die
Die Verantwortung für den Inhalt dieser Veröffentlichung trägt allein der Verfasser; die Kommission haftet nicht für die weitere Verwendung der darin enthaltenen Angaben.

Eine einfache SELECT-Anweisung würde zum Beispiel so aussehen:

SELECT spalte1, spalte2 FROM tabellen_name;

In dieser Anweisung ist "SELECT" das Schlüsselwort, "column1, column2" sind die Argumente und "table_name" ist die Tabelle, aus der die Daten abgerufen werden. Das Semikolon am Ende der Anweisung zeigt das Ende der SQL-Anweisung an.

2.3.2 SELECT-Anweisungen

Ein einfaches Beispiel für eine SELECT-Anweisung in Bezug auf unsere relationale Datenbank könnte wie folgt aussehen:

SELECT * **FROM** Mitarbeitende;

"**SELECT**" ist das Schlüsselwort und "
*" und "**FROM employees**" sind die Argumente

Diese SQL-Anweisung ruft alle Daten aus unserer Tabelle "Mitarbeitende" ab. Das * steht für alle und in diesem Fall für alle Spalten.

In den meisten Fällen wollen wir jedoch nicht alle gespeicherten Daten, sondern nur einen Teil davon. Zum Beispiel den Namen und die dazugehörige Telefonnummer aller Mitarbeitenden:

SELECT Name, TelNr **FROM** Mitarbeitende;

Betrachten wir nun die speziellen Argumente **FROM** und **WHERE:**

FROM bestimmt die Tabelle oder die Tabellen für diese Abfrage **WHERE** wird verwendet, um eine Bedingung zu unserer Abfrage hinzuzufügen; wenn die Bedingung ein Text ist, schließen wir den Text in einfache Anführungszeichen ein

Wenn wir z. B. nur die Telefonnummern unserer Mitarbeitenden mit dem Nachnamen Maus haben wollen, dann verwenden wir:

SELECT Name, TelNr **FROM** Mitarbeitende **WHERE** Name = 'Maus';

SELECT-Anweisungen sind sehr leistungsfähig und ermöglichen Data Scientists verschiedene Einblicke. Weitere spezielle Argumente findest du im Anhang zu diesem Skript.

Co-funded by the Erasmus+ Programme of the European Union

Mit Unterstützung des Programms Erasmus+ der Europäischen Union. Dieses Dokument und sein Inhalt spiegeln die
Die Verantwortung für den Inhalt dieser Veröffentlichung trägt allein der Verfasser; die Kommission haftet nicht für die weitere Verwendung der darin enthaltenen Angaben.

2.3.3 Andere nützliche SQL-Anweisungen

Die DML-Befehle (Data Manipulation Language) von SQL lauten SELECT, INSERT, UPDATE und DELETE und sind für die Verwaltung und Bearbeitung von Daten in einer relationalen Datenbank unerlässlich. Durch das Erlernen dieser 4 Befehle und ihrer effektiven Anwendung können Benutzer:innen SQL beherrschen und es für eine breite Palette von Datenbankaufgaben einsetzen.

SELECT: Der SELECT-Befehl wird verwendet, um Daten aus einer oder mehreren Tabellen in einer Datenbank abzurufen. Er ermöglicht es den Benutzer:innen, die Spalten anzugeben, die sie abrufen wollen, und die Daten nach bestimmten Kriterien zu filtern. Wir haben im letzten Kapitel mit ihnen gearbeitet.

INSERT: Der Befehl INSERT wird verwendet, um einer Tabelle in einer Datenbank neue Daten hinzuzufügen. Er ermöglicht es uns, die Werte, die in die Tabelle eingefügt werden sollen, und die Spalten, in die diese Werte eingefügt werden sollen, anzugeben.

> **INSERT INTO** tabelle_name (spalte1, spalte2 ...)
> **VALUES** (wert1, wert2 ... wertX);

Nach INSERT INTO folgt der Name der Tabelle, in die die Daten eingefügt werden sollen, im Beispiel "table_name".
Der nächste Teil der Anweisung gibt die Spalten der Tabelle an, in die die Daten eingefügt werden sollen, was durch "(column1, column2 ...)" dargestellt wird. Danach wird das Schlüsselwort VALUES verwendet, um die tatsächlichen Daten anzugeben, die in das Feld eingefügt werden sollen. Die Werte müssen mit der Reihenfolge der im vorherigen Abschnitt angegebenen Spalten übereinstimmen und werden durch "(value1, value2 ... valueX)" dargestellt.

Wenn wir zum Beispiel Daten in unsere Tabelle "Mitarbeitende" mit den Spalten "Name", "Adresse" und "TelNr" einfügen, könnte die SQL-Anweisung zum Einfügen eines neuen Datensatzes wie folgt aussehen:

INSERT INTO Mitarbeitende (Name, Adresse, TelNr) VALUES ('John Smith', 'High Street', 50573);

Beachte, dass nicht alle Spalten einer Tabelle in der INSERT INTO-Anweisung angegeben werden müssen. Wenn eine Spalte ausgelassen wird, weist die Datenbank entweder einen

Co-funded by the Erasmus+ Programme of the European Union
Mit Unterstützung des Programms Erasmus+ der Europäischen Union. Dieses Dokument und sein Inhalt spiegeln die
Die Verantwortung für den Inhalt dieser Veröffentlichung trägt allein der Verfasser; die Kommission haftet nicht für die weitere Verwendung der darin enthaltenen Angaben.

Standardwert zu oder fügt einen NULL-Wert in diese Spalte ein, je nachdem, wie die Tabelle erstellt wurde.

Außerdem muss die Reihenfolge, in der die Werte angegeben werden, mit der Reihenfolge der Spalten übereinstimmen. Wenn die Reihenfolge nicht übereinstimmt oder ein Wert fehlt, meldet die Datenbank einen Fehler.

UPDATE: Der UPDATE-Befehl wird verwendet, um vorhandene Daten in einer Tabelle in einer Datenbank zu ändern. Mit diesem Befehl können wir die neuen Werte angeben, welche die vorhandenen Werte ersetzen sollen, und die Daten nach bestimmten Kriterien zu filtern.

UPDATE tabelle_name
SET spalte1 = wert1, spalte2 = wert2 ... spalteN = wertN
[WHERE-Bedingung];

UPDATE table_name gibt den Namen der zu aktualisierenden Tabelle an.

SET column1 = value1, column2 = value2 ... columnN = valueN setzt die Werte einer oder mehrerer Spalten in der Tabelle auf neue Werte. Jede Spalte und ihr entsprechender Wert werden durch ein Gleichheitszeichen (=) getrennt, und mehrere Spalten werden durch Kommas getrennt.

[WHERE condition] ist eine optionale Klausel, die die Bedingung(en) angibt, die erfüllt sein müssen, damit die Aktualisierung stattfindet. Wenn keine Bedingung angegeben wird, werden alle Zeilen der Tabelle aktualisiert.

Beispiele für gültige Bedingungen:

```
    WHERE Name = 'Maus';
WHERE Alter > 50;
    WHERE Kategorie = 'Elektronik' AND stock_quantity > 0;
WHERE order_date < '2022-01-01';
```

DELETE: Der DELETE-Befehl wird verwendet, um Daten aus einer Tabelle in einer Datenbank zu entfernen. Er ermöglicht es uns, die Daten nach bestimmten Kriterien zu filtern und alle oder eine Teilmenge der Daten, die diesen Kriterien entsprechen, zu entfernen.

DELETE FROM table_name **WHERE** {condition};

Co-funded by the Erasmus+ Programme of the European Union

Mit Unterstützung des Programms Erasmus+ der Europäischen Union. Dieses Dokument und sein Inhalt spiegeln die
Die Verantwortung für den Inhalt dieser Veröffentlichung trägt allein der Verfasser; die Kommission haftet nicht für die weitere Verwendung der darin enthaltenen Angaben.

	DELETE FROM table_name gibt den Namen der Tabelle an, aus der Zeilen gelöscht werden sollen. WHERE {Bedingung} ist eine optionale Klausel, die die Bedingung(en) angibt, die erfüllt sein muss (müssen), damit die Zeilen gelöscht werden. Wenn keine Bedingung angegeben wird, werden alle Zeilen in der Tabelle gelöscht! Beispiele für gültige Bedingungen: **WHERE** NameID = 123456; Diese Anweisung würde die Zeile mit einer NameID von 12345 löschen **WHERE** NameID **IN** (12345, 23456, 34567); Diese Anweisung würde alle Zeilen löschen, deren "NameID" entweder 12345 oder 23456 oder 34567 ist. Bei der Verwendung der DELETE-Anweisung ist Vorsicht geboten, da sie Daten dauerhaft aus einer Tabelle entfernt. Es wird empfohlen, eine Sicherungskopie der Daten zu erstellen oder eine Transaktion zu verwenden, um die Änderungen bei Bedarf zurückzusetzen. Zusammenfassend lässt sich sagen, dass SQL eine leistungsstarke und flexible Datenbanksprache ist, die in vielen verschiedenen Branchen und Anwendungsbereichen eingesetzt wird. Ihre Benutzer:innenfreundlichkeit, Flexibilität und Effizienz machen sie zu einer hervorragenden Wahl für Unternehmen und Organisationen, die große Datenmengen verarbeiten und verwalten müssen. Mit seinen vielen Vorteilen und den verfügbaren Lernressourcen ist SQL ein wertvolles Werkzeug für jeden, der mit relationalen Datenbanken arbeitet.
Selbstbeurteilung (Multiple-Choice-Fragen und Antworten)	1. Wer verwendet SQL? A) Menschen, die mit Daten arbeiten, die in einer relationalen Datenbank gespeichert sind B) Web-Entwickler zur Programmierung von Websites C) Datenbankadministratoren zur Verwaltung der Daten 2. Wozu wird SQL verwendet? A) Anlegen und Löschen von Daten in einer Datenbank B) Für Zahlungen im Webshop C) Abrufen von Daten aus einer Nicht-Datenbankumgebung

Co-funded by the Erasmus+ Programme of the European Union

Mit Unterstützung des Programms Erasmus+ der Europäischen Union. Dieses Dokument und sein Inhalt spiegeln die
Die Verantwortung für den Inhalt dieser Veröffentlichung trägt allein der Verfasser; die Kommission haftet nicht für die weitere Verwendung der darin enthaltenen Angaben.

	3. Welche Schlüsselwörter können eine SQL-Anweisung einleiten? A) VON B) WO C) SELECT
Ressourcen (Videos, Verweislinks)	https://www.postgresql.org/ https://www.youtube.com/watch?v=HXV3zeQKqGY
Verwandtes Material	
Inhalt in 3 Ebenen gegliedert	**3. GitHub verstehen** **3.1 Zweck von GitHub** **3.1.1 Verwaltung von umfangreichen Software-Codes** Für alle Softwareentwickler:innen ist die Verwaltung des Codes ein wesentlicher Bestandteil der Programmierung. Versionskontrollsysteme ermöglichen es Entwickler:innen, ihre Codeänderungen zu kontrollieren, mit anderen zusammenzuarbeiten und ihre Projekte effizient zu verwalten. Ein beliebtes Versionskontrollsystem ist Git, und GitHub ist ein cloudbasierter Versionskontrolldienst für Softwareentwicklungsprojekte, der auf Git aufbaut. GitHub bietet eine benutzer:innenfreundliche Oberfläche zum Erstellen und Verwalten von Git-Repositories sowie Tools für die Zusammenarbeit mit anderen Entwicklern an einem Projekt. GitHub kann sowohl für persönliche als auch für kommerzielle Projekte genutzt werden und ist in der Basisversion kostenlos. GitHub kann als eine Art soziales Netzwerk für Softwareentwickler:innen bezeichnet werden. Mitglieder können einander folgen, die Arbeit anderer bewerten, Updates zu bestimmten Projekten erhalten und öffentlich oder privat kommunizieren. Es ermöglicht Entwickler:innen, ihre Arbeit zu teilen, von anderen zu lernen und eine Gemeinschaft aufzubauen. GitHub bietet auch eine umfassende Reihe von Projektmanagement-Tools, die es Entwickler:innen leicht machen, ihre Projekte zu verwalten. Dazu gehören Tools zum Verfolgen von Problemen, zum Organisieren von Aufgaben und zum Zusammenarbeiten mit anderen.

Co-funded by the Erasmus+ Programme of the European Union

Mit Unterstützung des Programms Erasmus+ der Europäischen Union. Dieses Dokument und sein Inhalt spiegeln die
Die Verantwortung für den Inhalt dieser Veröffentlichung trägt allein der Verfasser; die Kommission haftet nicht für die weitere Verwendung der darin enthaltenen Angaben.

GitHub lässt sich mit vielen anderen Tools und Diensten integrieren, darunter Dienste für die kontinuierliche Integration und Bereitstellung. Dies ermöglicht es Entwickler:innen, den Prozess des Testens, Erstellens und Bereitstellens ihrer Codeänderungen zu automatisieren.

Seit Ende 2018 ist GitHub Inc. im Besitz von Microsoft.

3.1.2 Was ist Versionskontrolle?

Bei der Arbeit an komplexen Softwareprojekten kann die Verfolgung von Änderungen und die Verwaltung mehrerer Codeversionen eine Herausforderung darstellen. Versionskontrollsysteme wie Git bieten eine Lösung für dieses Problem, indem sie die Änderungen verfolgen und aufzeichnen.

Git bietet Funktionen zum Wiederherstellen alter Versionen eines Projekts, Vergleichen, Analysieren, Zusammenführen von Änderungen und vieles mehr. Dieser Prozess wird **Versionskontrolle** genannt und hilft Entwickler:innen, den Überblick über die Geschichte ihres Codes zu behalten. Git ist nicht das einzige verfügbare Versionskontrollsystem. Andere Versionskontrollsysteme wie Perforce, Mercurial, CVS und SVN sind ebenfalls verfügbar. Aufgrund seiner Flexibilität, Geschwindigkeit und Benutzer:innenfreundlichkeit ist Git jedoch das beliebteste und am weitesten verbreitete Versionskontrollsystem unter Entwickler:innen.

Eines der einzigartigen Merkmale von Git ist, dass es sich um ein dezentrales Versionskontrollsystem handelt. Im Gegensatz zu anderen Systemen ist Git nicht auf einen zentralen Server angewiesen, um alte Versionen von Dateien aufzubewahren. Stattdessen arbeitet es vollständig lokal und speichert diese Daten in Ordnern auf der Festplatte einzelner Benutzer:innen. Dies wird als **Repository** bezeichnet. Dies ermöglicht es Entwickler:innen, offline an ihrem Code zu arbeiten, und erleichtert die Verfolgung von Änderungen, ohne auf einen zentralen Server angewiesen zu sein.

Wenn Benutzer:innen mit anderen an demselben Code arbeiten möchte, können sie eine Kopie ihres Repositorys online bereitstellen, auf die das gesamte Team zugreifen kann. So können mehrere Entwickler an der gleichen Codebasis arbeiten, ohne die Änderungen der anderen zu überschreiben.

Co-funded by the Erasmus+ Programme of the European Union

Mit Unterstützung des Programms Erasmus+ der Europäischen Union. Dieses Dokument und sein Inhalt spiegeln die
Die Verantwortung für den Inhalt dieser Veröffentlichung trägt allein der Verfasser; die Kommission haftet nicht für die weitere Verwendung der darin enthaltenen Angaben.

3.2 Wie man sie benutzt

3.2.1 Installieren Sie es auf Ihrem Gerät

Um Github produktiv nutzen zu können, benötigst du Git auf deinem lokalen Computer. Es gibt viele Git-Programme, die kostenlos oder kostenpflichtig genutzt werden können:

Windows - "Git für Windows" bietet einen GUI-Client und einen BASH-Befehlszeilenemulator.
https://git-scm.com/downloads/win oder
https://gitforwindows.org/

Linux - öffne einfach ein neues Terminal und installiere Git über den Paketmanager deiner Linux-Distribution. Für Ubuntu lautet der Befehl z. B. sudo apt-get install git

Mac OS - Der einfachste Weg ist, Homebrew zu installieren und dann einfach brew install git von deinem Terminal aus zu starten.
https://brew.sh/ oder https://git-scm.com/downloads/mac

3.2.2 Ein Konto erstellen und GitHub konfigurieren

Bevor du es verwenden kannst, musst du ein Github-Konto auf www.github.com erstellen.

Es gibt verschiedene Konfigurationen für das Aussehen und die Funktionalität des Clients. Entscheide nach deinen Vorlieben, um effizient zu arbeiten.

Wichtig ist jedoch die Konfiguration des **Benutzernamens** und der entsprechenden **E-Mail-Adresse**
öffne "Git Bash" und führe diese Befehle in der Befehlszeile aus:
git config --global user.name "Mary"
git config --global user.email "mary.smith@email.eu"

Alle Aktivitäten in Git sind mit dem jeweiligen Benutzernamen und der in der Konfiguration angegebenen E-Mail-Adresse verknüpft! Das macht Änderungen nachvollziehbar, weil andere Nutzer:innen immer wissen, wer die jeweiligen Änderungen vorgenommen hat, und schafft Übersicht in Projekten, an denen viele Entwickler:innen arbeiten.

3.3 Wie man GitHub verwendet

3.3.1 Allgemeine Begriffe in GitHub

Bei der Arbeit mit Git ist es wichtig, die Terminologie zu

Co-funded by the Erasmus+ Programme of the European Union

Mit Unterstützung des Programms Erasmus+ der Europäischen Union. Dieses Dokument und sein Inhalt spiegeln die
Die Verantwortung für den Inhalt dieser Veröffentlichung trägt allein der Verfasser; die Kommission haftet nicht für die weitere Verwendung der darin enthaltenen Angaben.

verstehen, um es effektiv nutzen zu können.

Ein **Repository**, oder kurz Repo, ist ein Ordner, in dem alle Dateien und ihre Versionshistorie gespeichert werden. Es ist der zentrale Ort zum Verwalten und Organisieren von Code. Ein Repository kann auf einem entfernten Server wie GitHub oder Bitbucket gehostet werden, oder es kann lokal auf Ihrem Computer gespeichert werden.

Ein **Branch** ist ein Arbeitsbereich, in dem du isolierte Änderungen vornehmen kannst, die sich nicht auf andere auswirken und eine eigene Historie haben. Er ist wie eine separate Zeitleiste, in der du mit Änderungen experimentieren kannst, ohne die Hauptcodebasis zu beeinträchtigen. Die Entwickler:innen können gleichzeitig an verschiedenen Branches arbeiten, was die Zusammenarbeit erleichtert und Konflikte vermeidet.

Ein **Commit** ist eine gespeicherte Aufzeichnung der Änderungen, die an einer Datei innerhalb des Projektarchivs vorgenommen wurden. Es handelt sich um den Zustand des Projektarchivs zu einem bestimmten Zeitpunkt. Ein Schnappschuss, zu dem der Code wiederhergestellt werden kann. Commits sind in Git unverzichtbar, da sie es ermöglichen, Änderungen im Laufe der Zeit zu verfolgen und bei Bedarf zu einer früheren Version des Codes zurückzukehren.

Nachdem Aktualisierungen an einem Repository vorgenommen wurden, können andere Entwickler:innen alle Änderungen mit einem **Pull Request** oder PR herunterladen. Eine Pull-Anfrage ist eine Anfrage, Änderungen von einem Branch in einen anderen zusammenzuführen. Dies ermöglicht es den Entwickler:innen, die Änderungen zu überprüfen und zu genehmigen, bevor sie in die Hauptcodebasis aufgenommen werden.

Ein Push ist der Prozess, bei dem eine lokale Änderung zum entfernten Repository hinzugefügt wird. Wenn du deine Änderungen pusht, werden sie für andere sichtbar und können heruntergeladen werden. Dies ist ein wichtiger Schritt in der Zusammenarbeit mit anderen Entwickler:innen.

Nachdem eine Pull-Anfrage genehmigt wurde, wird der Commit von einem Branch zum anderen **zusammengeführt**. Beim Merging werden die Änderungen von einem Branch in einen

Co-funded by the Erasmus+ Programme of the European Union

Mit Unterstützung des Programms Erasmus+ der Europäischen Union. Dieses Dokument und sein Inhalt spiegeln die
Die Verantwortung für den Inhalt dieser Veröffentlichung trägt allein der Verfasser; die Kommission haftet nicht für die weitere Verwendung der darin enthaltenen Angaben.

anderen, in der Regel den Haupt-Branch, zusammengeführt. Dieser Prozess stellt sicher, dass alle Änderungen in die Hauptcodebasis aufgenommen werden.

Ein **Klon** ist eine voll funktionsfähige Kopie eines Projekts. Das Kopieren eines Repositorys von einem entfernten Server ist ein Klonen. Auf diese Weise kannst du eine vollständige Kopie auf dem lokalen Rechner haben, was die Arbeit an deinem Code erleichtert, ohne auf eine Internetverbindung angewiesen zu sein.

3.3.2 Erste Schritte in Github

Wenn du Git verwendest, musst duzunächst einen Ordner erstellen

mkdir learning-git (=Verzeichnis erstellen)

in dem das Repository für jedes Programm oder Projekt erstellt werden kann

```
cd lernen-git          (= Verzeichnis wechseln)
git init
```

Anschließend kannst du ein lokales und ein Online-Repositorium verknüpfen mit

```
git remote add origin
https://github.com/[Nutzername]/[Projektname].git
```

Git arbeitet mit dem Konzept einer "Staging Area". Zu Beginn ist der Staging-Bereich leer. Dateien können mit dem Befehl git add hinzugefügt werden (oder sogar einzelne Zeilen und Teile von Dateien) und schließlich wird mit git commit alles übertragen (ein Schnappschuss erstellt):

```
git add learning_script.txt
git commit -m "firstsnapshot"
```

Um Code in ein entferntes Projektarchiv hochzuladen, verbinde dich sich zunächst mit diesem Speicherort

```
git remote add origin https://github.com/learning.git
```

Dann können die lokalen Commits auf den Server übertragen werden, was jedes Mal geschieht, wenn wir das entfernte Repository aktualisieren wollen

Co-funded by the Erasmus+ Programme of the European Union

Mit Unterstützung des Programms Erasmus+ der Europäischen Union. Dieses Dokument und sein Inhalt spiegeln die
Die Verantwortung für den Inhalt dieser Veröffentlichung trägt allein der Verfasser; die Kommission haftet nicht für die weitere Verwendung der darin enthaltenen Angaben.

git push server_origin local_master

Sobald dies geschehen ist, können andere Entwickler:innen die Änderungen aus dem entfernten Repository mit einem einzigen Befehl herunterladen

git pull origin master

Um ein ganzes Programm oder Projekt zu klonen, verwende

git clone https://github.com/tutorial.git

3.3.3 Branch-bezogene Befehle (branch)

Bei der Entwicklung einer neuen Funktion in der Softwareentwicklung ist es am besten, an einer Kopie des Originalprojekts, einem so genannten Branch (branch), zu arbeiten. Ein Branch ist eine separate Kopie der Codebasis, die es den Entwicklern ermöglicht, Änderungen vorzunehmen, ohne die Live-Version des Codes zu beeinflussen.

Jeder Branch hat seine eigene Historie und isoliert die an ihm vorgenommenen Änderungen von anderen Branches, bis beschlossen wird, sie zusammenzuführen. Die Vorteile dieses Ansatzes sind:

1. Eine stabile (Live-)Version des Codes ist nicht von unerwünschten Fehlern betroffen: Wenn du an einem separaten Branch arbeitest, wirken sich Fehler, die während des Entwicklungsprozesses auftreten, nicht auf die Live-Version des Codes aus.

2. Ein Team von Entwickler:innen kann an vielen Funktionen gleichzeitig arbeiten: Branches ermöglichen es mehreren Entwickler:innen, gleichzeitig an verschiedenen Funktionen zu arbeiten, ohne sich gegenseitig in die Arbeit zu verwickeln.

3. Jede:r Entwickler:in kann an dem eigenen Branch arbeiten, ohne Gefahr zu laufen, dass die Codebasis durch die Arbeit anderer Entwickler:innen verändert wird: Die Arbeit an separaten Branches stellt sicher, dass die Entwickler:innen unabhängig voneinander arbeiten können und Konflikte zwischen ihren Codeänderungen vermieden werden.

4. Mehrere Versionen desselben Features können in verschiedenen Branches entwickelt und dann verglichen werden, um die beste Version zu ermitteln: Branches ermöglichen es den Entwickler:innen, mit verschiedenen

Co-funded by the Erasmus+ Programme of the European Union

Mit Unterstützung des Programms Erasmus+ der Europäischen Union. Dieses Dokument und sein Inhalt spiegeln die
Die Verantwortung für den Inhalt dieser Veröffentlichung trägt allein der Verfasser; die Kommission haftet nicht für die weitere Verwendung der darin enthaltenen Angaben.

	Ansätzen für eine Funktion zu experimentieren und die Ergebnisse zu vergleichen, bevor die Änderungen in die Hauptcodebasis übernommen werden. Zusammenfassend lässt sich sagen, dass die Verwendung von Branches eine bewährte Praxis in der Softwareentwicklung ist, da sie dazu beiträgt, Konflikte und Fehler zu vermeiden, Teamarbeit ermöglicht und das Experimentieren und Vergleichen verschiedener Ansätze für eine Funktion erlaubt. Der Standard-Branch eines jeden Repositorys heißt master. Um weitere Branches zu erstellen, verwende den Befehl git branch <Name> Wechsle auf den neu erstellten Branch, indem du git checkout <Name> Um zwei Branches zusammenzuführen, wechsele zu einem und verwende git merge <Name> Um einen Branch zu löschen, verwende git branch -d <Name> Zusammenfassend lässt sich sagen, dass viele Unternehmen GitHub nutzen. Wenn du also einen Job suchst, bist du gut beraten, wenn du dich bereits mit GitHub auskennst. GitHub ist auch eine Lern- und Kooperationsplattform. Erforsche es und erweitere dein Wissen und deine Community.
Selbstbeurteilung (Multiple-Choice-Fragen und Antworten)	1. Warum ist Versionskontrolle wichtig? A) Schnelle Entwicklung von Code B) Entwickeln Sie ein Programm mit einem Team in einem nachvollziehbaren Prozess C) Erforderlich für den Verkauf des Projekts 2. Wer kann GitHub nutzen? A) Jeder mit einem Git-Konto B) Jeder, der es gekauft hat C) Nur geschulte Entwickler

Co-funded by the Erasmus+ Programme of the European Union

Mit Unterstützung des Programms Erasmus+ der Europäischen Union. Dieses Dokument und sein Inhalt spiegeln die
Die Verantwortung für den Inhalt dieser Veröffentlichung trägt allein der Verfasser; die Kommission haftet nicht für die weitere Verwendung der darin enthaltenen Angaben.

	3. Was ist ein Branch? A) Ein zugelassenes Unternehmen B) Ein spezifisches Projekt C) Eine Kopie des Originalprojekts
Ressourcen (Videos, Referenzlink)	https://www.github.com https://git-scm.com/downloads/winh ttps://gitforwindows.org/ https://brew.sh/ https://git-scm.com/downloads/mac
Verwandtes Material	
Verwandte PPT	
Literaturverzeichnis	
Zur Verfügung gestellt von	[Women in AI Austria]

Co-funded by the Erasmus+ Programme of the European Union

Mit Unterstützung des Programms Erasmus+ der Europäischen Union. Dieses Dokument und sein Inhalt spiegeln die
Die Verantwortung für den Inhalt dieser Veröffentlichung trägt allein der Verfasser; die Kommission haftet nicht für die weitere Verwendung der darin enthaltenen Angaben.