aggregatfunktionenSELECT <Aggregatfunktion>(Spalte) FROM Tabelle [WHERE Bedingung] [GROUP BY Spalte] [ORDER BY Spalte];

**1. Anzahl der Bücher:**

SELECT COUNT(\*) FROM buch;

**2. Anzahl der Mitglieder aus Appenzell:**

SELECT mitglied.ort AS Ort, COUNT(\*) AS Mitglieder

FROM mitglied

WHERE ort = "Appenzell"

GROUP BY Ort;

**3. Anzahl der Bücher je Genre (absteigend sortiert):**

SELECT genre, COUNT(\*) AS Anzahl

FROM buch

GROUP BY genre

ORDER BY COUNT(\*) DESC;

**4. Ausgeliehene Bücher je Mitglied (mit Vor- und Nachname):**

SELECT CONCAT(vorname, " ", name) AS Name, COUNT(\*) AS Anzahl

FROM mitglied

JOIN ausleihe ON mitglied.idmitglied = ausleihe.idmitglied

GROUP BY mitglied.idmitglied

ORDER BY Anzahl DESC;

**5. Autoren mit Anzahl der Bücher (mit Vor- und Nachname):**

SELECT CONCAT(vorname, " ", name) AS Name, COUNT(\*) AS "Anzahl Bücher"

FROM autor

JOIN autor\_buch ON autor.idautor = autor\_buch.id\_autor

GROUP BY autor.idautor

ORDER BY COUNT(\*) DESC;

**Aufbau von MySQL-Aggregatfunktionen:Grundstruktur:**SELECT <Aggregatfunktion>(Spalte) FROM Tabelle [WHERE Bedingung] [GROUP BY Spalte] [ORDER BY Spalte]; **Benutzer und rollen**

**Rollen**

///CREATE ROLE 'rollenname';delete role

// DROP ROLE 'rollenname';

// --

**Berechtigungen einer Rolle zuweisen**

GRANT SELECT, INSERT ON datenbank.tabelle TO 'rollenname'; --

**Eine Rolle einem Benutzer zuweisen**

GRANT 'rollenname' TO 'benutzername'@'host';// --

Berechtigungen von einer Rolle widerrufen

REVOKE SELECT, INSERT ON datenbank.tabelle FROM 'rollenname'; --

**Eine Rolle von einem Benutzer entfernen**

REVOKE 'rollenname' FROM 'benutzername'@'host';

// SET DEFAULT ROLE 'rollenname' TO 'benutzername'@'host';

//create user SET DEFAULT ROLE 'rollenname' TO 'benutzername'@'host';

//JOINS

//SELECT spalten FROM tabelle1 INNER JOIN tabelle2 ON tabelle1.gemeinsame\_spalte = tabelle2.gemeinsame\_spalte;

**Verwende INNER JOIN, wenn du nur die Datensätze zurückgeben möchtest, die in beiden Tabellen eine Übereinstimmung haben**

//SELECT spalten FROM tabelle1 LEFT JOIN tabelle2 ON tabelle1.gemeinsame\_spalte = tabelle2.gemeinsame\_spalte;

**Verwende LEFT JOIN, wenn du alle Datensätze aus der linken Tabelle zurückgeben möchtest, zusammen mit den übereinstimmenden Datensätzen (falls vorhanden) aus der rechten Tabelle. RIGHT JOIN (RIGHT OUTER JOIN)**

//SELECT spalten FROM tabelle1 RIGHT JOIN tabelle2 ON tabelle1.gemeinsame\_spalte = tabelle2.gemeinsame\_spalte;

**Verwende RIGHT JOIN, wenn du alle Datensätze aus der rechten Tabelle zurückgeben möchtest, zusammen mit den übereinstimmenden Datensätzen (falls vorhanden) aus der linken Tabelle.**

//stored procedures DELIMITER

//CREATE PROCEDURE procedure\_name (IN param1 datatype, OUT param2 datatype, ...) BEGIN -- SQL statements END

//DELIMITER ; DELIMITER

//CREATE PROCEDURE AddBook(IN title VARCHAR(255), IN author VARCHAR(255), IN pub\_year INT) BEGIN INSERT INTO

Books (Title, Author, PublicationYear) VALUES (title, author, pub\_year); END // DELIMITER ;

Aufrufen CALL procedure\_name(param1, param2, ...); zb

CALL AddBook('SQL Essentials', 'Jane Doe', 2021);

//verwaltigungen von berechtigungen

GRANT EXECUTE ON PROCEDURE AddBook TO 'username'@'localhost';

REVOKE EXECUTE ON PROCEDURE AddBook FROM 'username'@'localhost';

//Transaktionen

START TRANSACTION; -- SQL statements COMMIT; -- Zum Abschließen der Transaktion –

oder ROLLBACK; -- Zum Rückgängigmachen der Transaktion, falls ein Fehler auftritt zb START TRANSACTION;

INSERT INTO Orders (product\_id, quantity, order\_date) VALUES (1, 10, NOW()); UPDATE Products SET stock = stock - 10 WHERE id = 1; --

Wenn alles in Ordnung ist COMMIT; -- Wenn ein Fehler auftritt ROLLBACK;

//views CREATE VIEW view\_name AS SELECT column1, column2, ... FROM table\_name WHERE condition; zb CREATE VIEW CustomerView AS SELECT CustomerID, CustomerName, ContactName FROM Customers WHERE ContactName IS NOT NULL; SELECT \* FROM view\_name;---

SELECT \* FROM view\_name;--- DROP VIEW view\_name;