This guide will work on assumption that you passed D426 and know a little SQL from it.

***Study Material -*** ZYBooks, ZYBooks, and more ZYBooks. I know I'm sorry, but it's going to give you exactly what you need to pass and nothing else. Plus the test itself is done in ZYBooks so you're going to have to get used to how it responds.

Here is how I used it, I slogged my way through chapter 1 and 2 to brush up from D426. Then I started actually studying for the test. I did the green labs in chapters 2, 7 , and 8 about four times then chapters 7 and 8 five more times. I basically did the labs until I could do them by heart and could correlate what they were asking from me and what SQL statements should use with proper syntax.

This is the only class I've taken that I would recommend taking the PA enough to memorize it as the OA is absurdly close.

**Reference Material** - I used W3 for my syntax reference material for MYSQL statements while doing the labs until I was comfortable without it. This helped a lot as they just put the exact material on the left, where with ZYBooks you have to bounce around multiple pages to try and find the exact statement you're trying to write.

**Dissecting The ERROR Code -** You will notice sometimes when you submit a statement in ZYBooks it'll say something like SYNTAX ERROR blah blah look at MYSQL something. Most of it is unhelpful, scroll to the far right in the box (most of the time) and you will see it says something like REFERENCES something(something) or ID INT, this almost always means your syntax error is right before what this says. It is usually something like you forgot a , or you forgot to enclose (something) or other little mistake, but its almost always right before what it says the error is.

**How To Check Your Answer -** When you're doing the PA get used to checking your answers before you submit as the OA usually wont tell you if you did it correctly other than ERROR codes. I will try to break this down as best as I can.

* CHECK YOU SPELLING - Make sure you spelling and CAPS are correct, make sure that you did leave out an s at the end of something make sure its FirstName not firstname, it will kick back as wrong. Type things out exactly how they want them written ZyBooks hates you and has no sympathy if you type Id instead of ID or FirstNames instead of FirstName.
* Writing a SELECT - When asked to write a SELECT statement you just need to be sure what they ask for is displayed in the order they wanted it displayed, these questions are the easiest to check.
* INSERT INTO - INSERT INTO and any other statements where you are adding data to the tables can be checked by putting a SELECT \* FROM TableName; after the close; of your first statement. Verify that the data was added, then delete the SELECT statement, run it again to make sure and then submit.
* CREATE TABLE - CREATE TABLE or adding columns without data are the hardest to check because you haven't been taught how in this class or the last class. A SELECT statement will return nothing, This is where you learn the DESCRIBE statement. After you close your statement ; write a second statement DESCRIBE TableName; You will see the columns you created and constraints assigned to them EX. INT, VARCHAR, CHAR.
* PRIMARY AND FOREIGN KEY - To make sure you've assigned the key correctly you can use the DESCRIBE statement just like before, I think it say PRI for PRIMARY and MUL for FOREIGN. This is to make sure you've assigned the key to the correct table/column I used it mostly for FOREIGN KEY as I had a tendency to reverse it somehow.

I started this class in earnest on Sunday morning and tested out Tuesday night using this method, I found the biggest help was when I discovered the DESCRIBE statement as I was able to verify my statements worked as intended before submitting. I sat on this class for weeks before actually giving it a go because D426 burnt me out so hard but this class is so much better as I feel like I actually did learn something from it.

**UPDATE employee**

**SET salary = salary + 1000**

**WHERE salary < 70000;**

**San Francisco, CA 94110, USA**

**How many attributes are present in the address fragment?**

4

Each entity is considered an attribute.  
-San Francisco  
-CA  
-94110  
-USA

**The Package table has the following columns:  
Weight  
Description  
LastChangedDate  
TrackingNumber  
Which column should be designated the primary key for the Package table?**

TrackingNumber.

This is the only option that will always have to be UNIQUE.

**Which data type will store "2022-01-10 14:22:12" as a temporal value without loss of information?**

Date maybe timestamp

**Which SQL command is an example of data definition language (DDL)?**  
UPDATE  
**ALTER**  
SELECT  
DELETE

**How would a database engine process an update that violates a RESTRICT referential integrity constraint?**

?

**Which restriction applies when using a materialized view?**

A materialized view is a pre-computed data set derived from a query specification (the SELECT in the view definition) and stored for later use.  
  
- Data is pre-computed, so querying a materialized view is faster than executing a query against the base table of the view. This performance difference can be significant when a query is run frequently or is sufficiently complex. As a result, materialized views can speed up expensive aggregation, projection, and selection operations, especially those that run frequently and that run on large data sets.

**7.1 LAB - Alter Movie table**

The **Movie** table has the following columns:

* **ID** - positive integer
* **Title** - variable-length string
* **Genre** - variable-length string
* **RatingCode** - variable-length string
* **Year** - integer

Write ALTER statements to make the following modifications to the Movie table:

1. Add a **Producer** column with VARCHAR data type (max 50 chars).
2. Remove the **Genre** column.
3. Change the Year column's name to **ReleaseYear**, and change the data type to **SMALLINT**.

**Answer**

ALTER TABLE Movie ADD Producer VARCHAR(50);

ALTER TABLE Movie DROP COLUMN Genre;

ALTER TABLE Movie CHANGE COLUMN Year ReleaseYear SMALLINT;

**7.2 LAB - Insert rows into Horse table**

The **Horse** table has the following columns:

* **ID** - integer, auto increment, primary key
* **RegisteredName** - variable-length string
* **Breed** - variable-length string, must be one of the following: Egyptian Arab, Holsteiner, Quarter Horse, Paint, Saddlebred
* **Height** - decimal number, must be between 10.0 and 20.0
* **BirthDate** - date, must be on or after Jan 1, 2015

Insert the following data into the Horse table:

| RegisteredName | Breed | Height | BirthDate |
| --- | --- | --- | --- |
| Babe | Quarter Horse | 15.3 | 2015-02-10 |
| Independence | Holsteiner | 16.0 | 2017-03-13 |
| Ellie | Saddlebred | 15.0 | 2016-12-22 |
| *NULL* | Egyptian Arab | 14.9 | 2019-10-12 |

**Answer**

INSERT INTO Horse (RegisteredName, Breed, Height, BirthDate) VALUES

('Babe', 'Quarter Horse', 15.3, '2015-02-10'),

('Independence', 'Holsteiner', 16.0, '2017-03-13'),

('Ellie', 'Saddlebred', 15.0, '2016-12-22'),

(NULL, 'Egyptian Arab', 14.9, '2019-10-12');

**7.3 LAB - Update rows in Horse table**

The **Horse** table has the following columns:

* **ID** - integer, auto increment, primary key
* **RegisteredName** - variable-length string
* **Breed** - variable-length string, must be one of the following: Egyptian Arab, Holsteiner, Quarter Horse, Paint, Saddlebred
* **Height** - decimal number, must be ≥ 10.0 and ≤ 20.0
* **BirthDate** - date, must be ≥ Jan 1, 2015

Make the following updates:

1. Change the height to **15.6** for horse with ID 2.
2. Change the registered name to **Lady Luck** and birth date to **May 1, 2015** for horse with ID 4.
3. Change every horse breed to ***NULL*** for horses born on or after December 22, 2016.

**Answer**

UPDATE Horse

SET Height = 15.6

WHERE ID = 2;

UPDATE Horse

SET RegisteredName = 'Lady Luck', BirthDate = '2015-05-01'

WHERE ID = 4;

UPDATE Horse

SET Breed = NULL

WHERE BirthDate >= '2016-12-22';

**7.4 LAB - Delete rows from Horse table**

The **Horse** table has the following columns:

* **ID** - integer, auto increment, primary key
* **RegisteredName** - variable-length string
* **Breed** - variable-length string
* **Height** - decimal number
* **BirthDate** - date

Delete the following rows:

1. Horse with ID 5.
2. All horses with breed Holsteiner or Paint.
3. All horses born before March 13, 2013.

**Answer**

DELETE FROM Horse

WHERE ID = 5;

DELETE FROM Horse

WHERE Breed IN ('Holsteiner', 'Paint');

DELETE FROM Horse

WHERE BirthDate < '2013-03-13';

**7.5 LAB - Select horses with logical operators**

The **Horse** table has the following columns:

* **ID** - integer, primary key
* **RegisteredName** - variable-length string
* **Breed** - variable-length string
* **Height** - decimal number
* **BirthDate** - date

Write a SELECT statement to select the registered name, height, and birth date for only horses that have a height between 15.0 and 16.0 (inclusive) or have a birth date on or after January 1, 2020.

Answer

SELECT RegisteredName, Height, BirthDate

FROM Horse

WHERE (Height BETWEEN 15.0 AND 16.0)

OR (BirthDate >= '2020-01-01');

**7.6 LAB - Create Movie table**

Create a **Movie** table with the following columns:

* **ID** - positive integer with maximum value of 50,000
* **Title** - variable-length string with up to 50 characters
* **Rating** - fixed-length string with 4 characters
* **ReleaseDate** - date
* **Budget** - decimal value representing a cost of up to 999,999 dollars, with 2 digits for cents

**Answer**

CREATE TABLE Movie (

ID SMALLINT UNSIGNED PRIMARY KEY CHECK (ID <= 50000),

Title VARCHAR(50),

Rating CHAR(4),

ReleaseDate DATE,

Budget DECIMAL(8,2) CHECK (Budget <= 999999.99)

);

**7.7 LAB - Create Student table with constraints**

Create a **Student** table with the following column names, data types, and constraints:

* **ID** - integer with range 0 to 65 thousand, auto increment, primary key
* **FirstName** - variable-length string with max 20 chars, not NULL
* **LastName** - variable-length string with max 30 chars, not NULL
* **Street** - variable-length string with max 50 chars, not NULL
* **City** - variable-length string with max 20 chars, not NULL
* **State** - fixed-length string of 2 chars, not NULL, default "TX"
* **Zip** - integer with range 0 to 16 million, not NULL
* **Phone** - fixed-length string of 10 chars, not NULL
* **Email** - variable-length string with max 30 chars, must be unique

**Answer**

CREATE TABLE Student (

ID SMALLINT UNSIGNED AUTO\_INCREMENT PRIMARY KEY,

FirstName VARCHAR(20) NOT NULL,

LastName VARCHAR(30) NOT NULL,

Street VARCHAR(50) NOT NULL,

City VARCHAR(20) NOT NULL,

State CHAR(2) NOT NULL DEFAULT 'TX',

Zip MEDIUMINT UNSIGNED NOT NULL,

Phone CHAR(10) NOT NULL,

Email VARCHAR(30) UNIQUE

);

**7.8 LAB - Create Horse table with constraints**

Create a **Horse** table with the following columns, data types, and constraints. NULL is allowed unless 'not NULL' is explicitly stated.

* **ID** - integer with range 0 to 65535, auto increment, primary key
* **RegisteredName** - variable-length string with max 15 chars, not NULL
* **Breed** - variable-length string with max 20 chars, must be one of the following: Egyptian Arab, Holsteiner, Quarter Horse, Paint, Saddlebred
* **Height** - number with 3 significant digits and 1 decimal place, must be ≥ 10.0 and ≤ 20.0
* **BirthDate** - date, must be ≥ Jan 1, 2015

**Answer**

CREATE TABLE Horse (

ID SMALLINT UNSIGNED AUTO\_INCREMENT PRIMARY KEY,

RegisteredName VARCHAR(15) NOT NULL,

Breed VARCHAR(20) CHECK (Breed IN ('Egyptian Arab', 'Holsteiner', 'Quarter Horse', 'Paint', 'Saddlebred')),

Height DECIMAL(3,1) CHECK (Height >= 10.0 AND Height <= 20.0),

BirthDate DATE CHECK (BirthDate >= '2015-01-01')

);

**7.9 LAB - Create LessonSchedule table with FK constraints**

Two tables are created:

1. **Horse** with columns:
   * **ID** - integer, primary key
   * **RegisteredName** - variable-length string
2. **Student** with columns:
   * **ID** - integer, primary key
   * **FirstName** - variable-length string
   * **LastName** - variable-length string

Create the **LessonSchedule** table with columns:

* **HorseID** - integer with range 0 to 65 thousand, not NULL, partial primary key, foreign key references Horse(ID)
* **StudentID** - integer with range 0 to 65 thousand, foreign key references Student(ID)
* **LessonDateTime** - date/time, not NULL, partial primary key

If a row is deleted from Horse, the rows with the same horse ID should be deleted from LessonSchedule automatically.

If a row is deleted from Student, the same student IDs should be set to NULL in LessonSchedule automatically.

**Answer**

CREATE TABLE LessonSchedule (

HorseID SMALLINT UNSIGNED NOT NULL,

StudentID SMALLINT UNSIGNED,

LessonDateTime DATETIME NOT NULL,

PRIMARY KEY (HorseID, LessonDateTime),

FOREIGN KEY (HorseID) REFERENCES Horse(ID) ON DELETE CASCADE,

FOREIGN KEY (StudentID) REFERENCES Student(ID) ON DELETE SET NULL

);

**7.10 LAB - Rollback and savepoint (Sakila)**

Refer to the actor table of the Sakila database. The table in this lab has the same columns and data types but fewer rows.

Start a transaction and:

1. Insert a new actor with values 999, 'NICOLE', 'STREEP', '2021-06-01 12:00:00'
2. Set a SAVEPOINT.
3. Delete the actor with first name 'CUBA'.
4. Select all actors.
5. Roll back to the savepoint.
6. Select all actors a second time.

The actor with first name 'CUBA' should appear in the second SELECT but not the first.

**Answer**

-- Your SQL statements go here

-- Start the transaction

START TRANSACTION;

-- 1. Insert a new actor

INSERT INTO actor (actor\_id, first\_name, last\_name, last\_update)

VALUES (999, 'NICOLE', 'STREEP', '2021-06-01 12:00:00');

-- 2. Set a SAVEPOINT

SAVEPOINT my\_savepoint;

-- 3. Delete the actor with first name 'CUBA'

DELETE FROM actor WHERE first\_name = 'CUBA';

-- 4. Select all actors

SELECT \* FROM actor;

-- 5. Roll back to the savepoint

ROLLBACK TO my\_savepoint;

-- 6. Select all actors a second time

SELECT \* FROM actor;

-- End the transaction (optional, based on the desired outcome)

COMMIT;

**D427 Practice Test ANSWER KEY**

1. Seattle, WA 98111

USA

How many attributes are present in the address fragment?

1. 1
2. 2
3. 3
4. 4
5. The Book table has the following columns:

genre – varchar(20)

pages – integer

author\_id – char(3)

isbn\_number – varchar(20)

Which column should be designated at the primary key for the Book table?

1. genre
2. pages
3. author\_id
4. isbn\_number
5. The Book table has the following columns:

genre – varchar(20)

pages – integer

author\_id – char(3)

isbn\_number – varchar(20)

Which column should be designated as the foreign key for the Book table?

1. genre
2. pages
3. author\_id
4. isbn\_number
5. Which data type represents numbers with fractional values:
6. Integer
7. Decimal
8. Character
9. Binary
10. Which of the following are DDL commands?
11. INSERT
12. SELECT
13. CREATE INDEX
14. UPDATE
15. Which of the following is a DML command?
16. CREATE VIEW
17. CREATE TABLE
18. INSERT
19. ALTER INDEX

CREATE TABLE Invoice (

invoice\_id INT NOT NULL AUTO\_INCREMENT,

date DATE NOT NULL,

customer\_id INT NOT NULL,

PRIMARY KEY (invoice\_id),

FOREIGN KEY (customer\_id) REFERENCES Customer (customer\_id) **ON DELETE CASCADE**

);

Looking at the Customer and Invoice tables and the CREATE TABLE for the Invoice table with foreign key reference statement above, what would happen to invoices in the Invoice table that are linked to a customer if that customer is deleted.

1. Those invoices would remain in the database.
2. Those invoices would be deleted also.
3. The Customer ID for those invoices would be changed to NULL.
4. Nothing would happen.

Invoice ID (PK)

Date

Customer ID (FK)

Customer ID (PK)

Customer Last Name

Customer First Name

Street Address

City

State

Zip

CREATE TABLE Invoice (

invoice\_id INT NOT NULL AUTO\_INCREMENT,

date DATE NOT NULL,

customer\_id INT NOT NULL,

PRIMARY KEY (invoice\_id),

FOREIGN KEY (customer\_id) REFERENCES Customer (customer\_id) **ON DELETE RESTRICT**

);

Looking at the Customer and Invoice tables and the CREATE TABLE for the Invoice table with foreign key reference statement above, what would happen to invoices in the Invoice table that are linked to a customer if that customer is deleted.

1. Those invoices would remain in the database.
2. Those invoices would be deleted also.
3. The Customer ID for those invoices would be changed to NULL.
4. The delete of the Customer would not be allowed.

Invoice ID (PK)

Date

Customer ID (FK)

Customer ID (PK)

Customer Last Name

Customer First Name

Street Address

City

State

Zip

CREATE TABLE Invoice (

invoice\_id INT NOT NULL AUTO\_INCREMENT,

date DATE NOT NULL,

customer\_id INT NOT NULL,

PRIMARY KEY (invoice\_id),

FOREIGN KEY (customer\_id) REFERENCES Customer (customer\_id) **ON DELETE SET TO NULL**

);

Looking at the Customer and Invoice tables and the CREATE TABLE for the Invoice table with foreign key reference statement above, what would happen to invoices in the Invoice table that are linked to a customer if that customer is deleted.

1. Those invoices would remain in the database.
2. Those invoices would be deleted also.
3. The Customer ID for those invoices would be changed to NULL.
4. The delete of the Customer would not be allowed.

Invoice ID (PK)

Date

Customer ID (FK)

Customer ID (PK)

Customer Last Name

Customer First Name

Street Address

City

State

Zip

10.

Which of the following are true about materialized view (Choose 2)?

1. It is a base table.
2. It is stored.
3. It must be refreshed whenever the base table changes.
4. The results are stored as a temporary table.

11.

The Customer table will have the following columns:  
CustomerID—positive integer  
FirstName—variable-length string with up to 50 characters  
MiddleInitial—fixed-length string with 1 character  
LastName—variable-length string with up to 50 characters  
DateOfBirth—date  
CreditLimit—positive decimal value representing a cost of up to $19,999, with 2 digits for cents  
Write a SQL statement to create the Customer table.  
Do not add any additional constraints to any column beyond what is stated.

CREATE TABLE Customer (

CustomerID INT UNSIGNED,

FirstName VARCHAR(50),

MiddleInitial CHAR(1),

LastName VARCHAR(50),

DateOfBirth DATE,

CreditLimit DECIMAL(7,2) UNSIGNED

);

12.

The Genre table has the following columns:  
GenreCode—variable-length string, primary key  
GenreDescription—variable-length string

The Book table should have the following columns:  
Title—variable-length string, maximum 30 characters  
GenreCode—variable-length string, maximum 5 characters

Write a SQL statement to create the Book table. Designate the GenreCode column in the Book table as a foreign key to the GenreCode column in the Genre table.

**CREATE TABLE Book (**

**Title VARCHAR(30),**

**GenreCode VARCHAR(5),**

**FOREIGN KEY (GenreCode) REFERENCES Genre(GenreCode)**

**);**

13.

The Automobile table has the following columns:

ID—integer, primary key  
Make—variable-length string  
Model—variable-length string  
Year—integer

A new column must be added to the Automobile table:  
Column name: SafetyRating  
Data type: decimal(3,1)

Write a SQL statement to add the SafetyRating column to the Automobile table.

**ALTER TABLE Automobile**

**ADD SafetyRating DECIMAL(3,1);**

14.

The Book table has the following columns:  
ID—integer, primary key  
Title—variable-length string  
Genre—variable-length string  
Year—integer

Write a SQL statement to create a view named MyBooks that contains the Title, Genre, and Year columns for all ~~movies~~ books. Ensure your result set returns the columns in the order indicated.

**CREATE VIEW MyBooks AS**

**SELECT Title, Genre, Year**

**FROM Book;**

15.

A database has a view named BookView.  
Write a SQL statement to delete the view named BookView from the database.

**DROP VIEW BookView;**

16.

The Book table has the following columns:  
ID—integer, primary key  
Title—variable-length string  
Genre—variable-length string  
Year—integer

Write a SQL statement to modify the Book table to make the ID column the primary key.

**ALTER TABLE Book**

**ADD PRIMARY KEY (ID);**

17.

The Book table has the following columns:  
ID—integer, primary key  
Title—variable-length string  
Genre—variable-length string  
Year—integer

The YearSales table has the following columns:  
Year—integer  
TotalSales—bigint unsigned  
Releases—integer

Write a SQL statement to designate the Year column in the Book table as a foreign key to the Year column in the TotalSales table.

**ALTER TABLE Book**

**ADD FOREIGN KEY (Year) REFERENCES YearSales (Year);**

18.

The Book table has the following columns:  
ID—integer, primary key  
Title—variable-length string  
Genre—variable-length string  
Year—integer  
Write a SQL statement to create an index named idx\_year on the Year column of the Book table.

**CREATE INDEX idx\_year ON Book(Year);**

19.

The Book table has the following columns:  
ID—integer, primary key, auto\_increment  
Title—variable-length string  
Genre—variable-length string  
Year—integer

The following data needs to be added to the Book table:  
Title Genre Year  
The Joy Luck Club Fiction 1989

Write a SQL statement to insert the indicated data into the Book table.

**INSERT INTO Book (Title, Genre, Year) VALUES**

**(‘The Joy Luck Club’, ‘Fiction’, 1989);**

20.

The Book table has the following columns:  
ID—integer, primary key, auto\_increment  
Title—variable-length string  
Genre—variable-length string  
Year—integer

Write a SQL statement to delete the row with the ID value of 3 from the Book table.

**DELETE from Book WHERE ID = 3;**

21.

The Book table has the following columns:  
ID—integer, primary key, auto\_increment  
Title—variable-length string  
Genre—variable-length string  
Year—integer

Write a SQL statement to update the Year value to be 2022 for all books with a Year value of 2020.

**UPDATE Book  
SET Year = 2022  
WHERE Year = 2020;**

22. Which query illustrates performing an outer join of the Movie table with a different table?

1. SELECT B.Title, A.Author FROM Book B, Author A  
   WHERE B.AuthorID = A.AuthorID;
2. SELECT B.Title, A.Author FROM Book B  
   LEFT JOIN Book MB ON B.ID = MB.IF, Author A
3. SELECT Book.Title, A.Author FROM Book B  
   RIGHT JOIN Author A ON B.AuthorID = A.ID
4. SELECT B.Title, A.Author FROM Book B  
   INNER JOIN Author A ON B.AuthorID = A.ID

This question is asking which statement illustrates an OUTER JOIN, not an INNER JOIN. The only 2 statements above that include outer joins are b and c (LEFT JOIN and RIGHT JOIN).

B is an invalid statement, so C has to be the answer. Please review this document to get a good understanding of outer versus inner joins:

[Left joins / Right Joins / Inner Joins - Examples](https://westerngovernorsuniversity-my.sharepoint.com/:w:/g/personal/maria_schenk_wgu_edu/Eafty6p0RX9PhR5ytuM4igUB1ykNDPOm3eurRHGlA0U7HQ?e=R5Ujvj)

23.

Assume there are two tables, A and B.  
Which rows will always be included in the result set if Table A is inner joined with Table B?

a. Only rows in Tables A and B that share the join condition

b. All rows in Table B

c. All rows in Table A

d. Only rows in Tables A and B that do not share the join condition.

24.

The database contains a table named Book.  
Write a SQL query to return all data from the Book table without directly referencing any column names.

**SELECT \* FROM Book;**

25.

The Book table has the following columns:  
ID—integer, primary key, auto\_increment  
Title—variable-length string  
Genre—variable-length string  
Year—integer

Write a SQL query to retrieve the Title and Genre values for all records in the Book table with a Year value of 2020. Ensure your result set returns the columns in the order indicated.

**SELECT Title, Genre**

**FROM Book**

**WHERE Year = 2020;**

26.

The Book table has the following columns:  
ID—integer, primary key, auto\_increment  
Title—variable-length string  
Genre—variable-length string  
Year—integer

Write a SQL query to display all Title values in alphabetical order A–Z.

This question is testing to see if the student knows how to use the ORDER BY clause. Here is the general syntax:

A group of black text

Description automatically generated

And here is the correct answer.

**SELECT Title**

**FROM Book**

**ORDER BY Title ASC;**

Or, this is also correct because ascending order is the default.

**SELECT Title**

**FROM Book**

**ORDER BY Title;**

27.

The Book table has the following columns:  
ID—integer, primary key, auto\_increment  
Title—variable-length string  
Genre—variable-length string  
Year—integer

Write a SQL query to output the unique Genre values and the number of books with each genre value from the Book table as GenreCount. Sort the results by the Genre in alphabetical order A–Z. Ensure your result set returns the columns in the order indicated.

In this case you are being asked to count the number of Movies having each type of Genre. For this reason, you must use GROUP BY to group all moves with each Genre together and then do **a count. The attribute you are grouping on must be listed first.**

**SELECT Genre, COUNT(\*) AS GenreCount  
FROM Book  
GROUP BY Genre  
ORDER BY Genre ASC;**

Please take a look at these resources:

[GROUP BY](https://www.w3schools.com/sql/sql_groupby.asp)   
[COUNT](https://www.w3schools.com/sql/sql_count_avg_sum.asp)

28.

The Book table has the following columns:  
ID—integer, primary key, auto\_increment  
Title—variable-length string  
Genre—variable-length string  
Year—integer

The YearSales table has the following columns:  
Year—integer  
TotalSales—bigint unsigned  
Releases—integer  
Write a SQL query to display both the Title and the TotalSales (if available) for all books. Ensure your result set returns the columns in the order indicated.

Here, you must have a JOIN because you are pulling a value from each of the tables.

**SELECT Title, TotalSales   
FROM Book LEFT JOIN YearSales  
ON Book.Year = YearSales.Year;**

29.

The Book table has the following columns:  
ID—integer, primary key, auto\_increment  
Title—variable-length string  
Genre—variable-length string  
Year—integer

Write a SQL query to return how many books have a Year value of 2019.

**SELECT COUNT(\*)**

**FROM Book**

**WHERE Year = 2019;**