# Create Database

CREATE DATABASE IF NOT EXISTS student;

USE student;

# Create Table Student\_Personal\_Info

CREATE TABLE IF NOT EXISTS Student\_Personal\_Info (

student\_id INT PRIMARY KEY,

first\_name VARCHAR(50),

last\_name VARCHAR(50),

dob DATE,

address VARCHAR(100)

);

# Create Table Student\_Academic\_Info

CREATE TABLE IF NOT EXISTS Student\_Academic\_Info (

student\_id INT PRIMARY KEY,

subject VARCHAR(50),

marks INT,

grade CHAR(1)

);

# Insert 5 Records into Student\_Personal\_Info

INSERT INTO Student\_Personal\_Info (student\_id, first\_name, last\_name, dob, address)

VALUES

(1, 'Aarav', 'Sharma', '1990-01-01', '123 Main St'),

(2, 'Ananya', 'Patel', '1992-05-15', '456 Oak St'),

(3, 'Aryan', 'Gupta', '1993-08-20', '789 Pine St'),

(4, 'Ishita', 'Singh', '1995-02-10', '101 Elm St'),

(5, 'Rahul', 'Verma', '1997-04-25', '202 Maple St');

# Insert 5 Records into Student\_Academic\_Info

INSERT INTO Student\_Academic\_Info (student\_id, subject, marks, grade)

VALUES

(1, 'Math', 85, 'A'),

(2, 'English', 92, 'A'),

(3, 'Science', 78, 'B'),

(4, 'History', 95, 'A'),

(5, 'Computer Science', 88, 'B');

# ALTER table add foreign key

[ALTER](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/alter-table.html) [TABLE](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/alter-table.html) `student\_academic\_info` ADD CONSTRAINT `fk\_student\_id` FOREIGN KEY (`student\_id`) REFERENCES `student\_personal\_info`(`student\_id`) ON [DELETE](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/delete.html) RESTRICT ON [UPDATE](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/update.html) RESTRICT;

# Add a new column

ALTER TABLE Student\_Personal\_Info

ADD COLUMN email VARCHAR(100);

## Drop the 'address' column

ALTER TABLE Student\_Personal\_Info

DROP COLUMN address;

## Rename the 'dob' column

ALTER TABLE Student\_Personal\_Info

CHANGE COLUMN dob birthdate DATE;

## Modify the datatype of 'birthdate' column

ALTER TABLE Student\_Personal\_Info

MODIFY COLUMN birthdate DATETIME;

# Update the address for a specific student in Student\_Personal\_Info

UPDATE Student\_Personal\_Info

SET address = 'New Address'

WHERE student\_id = 1;

# Update the birthdate for a specific student in Student\_Personal\_Info

UPDATE Student\_Personal\_Info

SET birthdate = '1991-02-15'

WHERE student\_id = 2;

# Update the marks for a specific student and subject in Student\_Academic\_Info

UPDATE Student\_Academic\_Info

SET marks = 90

WHERE student\_id = 3 AND subject = 'Math';

# Update the grade for a specific student and subject in Student\_Academic\_Info

UPDATE Student\_Academic\_Info

SET grade = 'A+'

WHERE student\_id = 4 AND subject = 'History';

# Increase marks for all students in Computer Science in Student\_Academic\_Info

UPDATE Student\_Academic\_Info

SET marks = marks + 5

WHERE subject = 'Computer Science';

# Update email for a specific student in Student\_Personal\_Info based on a pattern

UPDATE Student\_Personal\_Info

SET email = CONCAT(first\_name, '\_', last\_name, '@newdomain.com')

WHERE student\_id = 5;

# Display the updated records in Student\_Personal\_Info

SELECT \* FROM Student\_Personal\_Info;

# Display the updated records in Student\_Academic\_Info

SELECT \* FROM Student\_Academic\_Info;

# Display the initial records in Student\_Personal\_Info

SELECT \* FROM Student\_Personal\_Info;

# Truncate the Student\_Personal\_Info table

TRUNCATE TABLE Student\_Personal\_Info;

# Display the records after truncation

SELECT \* FROM Student\_Personal\_Info;

# COMMIT or ROLLBACK

-- Start a transaction explicitly

START TRANSACTION;

-- Update a record in Student\_Personal\_Info

UPDATE Student\_Personal\_Info

SET address = 'New Address'

WHERE student\_id = 1;

-- Insert a new record in Student\_Academic\_Info

INSERT INTO Student\_Academic\_Info (student\_id, subject, marks, grade)

VALUES (1, 'History', 88, 'B');

-- Commit the transaction to make the changes permanent

COMMIT;

# ROLLBACK

-- Start a transaction explicitly

START TRANSACTION;

-- Update a record in Student\_Personal\_Info

UPDATE Student\_Personal\_Info

SET address = 'New Address'

WHERE student\_id = 1;

-- Insert a new record in Student\_Academic\_Info

INSERT INTO Student\_Academic\_Info (student\_id, subject, marks, grade)

VALUES (1, 'History', 88, 'B');

-- Rollback the transaction to undo the changes

ROLLBACK;

# Aggregate functions in SQL

1. **COUNT():** Returns the number of rows in a set.
2. **SUM():** Returns the sum of values in a set.
3. **AVG():** Returns the average of values in a set.
4. **MIN():** Returns the minimum value in a set.
5. **MAX():** Returns the maximum value in a set.

The **AS** clause in SQL is used to give a column or table a temporary alias, or a different name, in the result set of a query.

SELECT first\_name SELECT first\_name AS "First Name", last\_name AS "Last Name"

FROM Student\_Personal\_Info; "First Name", last\_name AS "Last Name"

FROM Student\_Personal\_Info;

Example 1: Count the number of students in Student\_Personal\_Info

SELECT COUNT(\*) AS total\_students

FROM Student\_Personal\_Info;

Example 2: Calculate the average marks in Student\_Academic\_Info

SELECT AVG(marks) AS average\_marks

FROM Student\_Academic\_Info;

Example 3: Find the highest and lowest marks in Student\_Academic\_Info

SELECT MAX(marks) AS highest\_marks, MIN(marks) AS lowest\_marks

FROM Student\_Academic\_Info;

Example 4: Calculate the total number of students and the average marks in a specific subject

SELECT COUNT(\*) AS total\_students, AVG(marks) AS average\_marks

FROM Student\_Academic\_Info

WHERE subject = 'Math';

# **GROUP BY** clause in SQL is used to group rows that have the same values in specified columns into summary rows, like those returned by aggregate functions.

**SELECT column1, column2, ..., aggregate\_function(columnX)**

**FROM table**

**GROUP BY column1, column2, ...;**

breakdown of the key components:

* SELECT clause: Specifies the columns you want to display in the result set.
* FROM clause: Specifies the table from which to retrieve data.
* GROUP BY clause: Lists the columns based on which you want to group the data.
* Aggregate function(s): Perform calculations on the grouped data, like counting, summing, averaging, finding minimum or maximum, etc.

-- Example: Count the number of students in each subject

SELECT subject, COUNT(\*) AS total\_students

FROM Student\_Academic\_Info

GROUP BY subject;

# The **HAVING** clause in SQL is used in conjunction with the GROUP BY clause to filter the results of a grouped query based on specified conditions.

SELECT column1, column2, ..., aggregate\_function(columnX)

FROM table

GROUP BY column1, column2, ...

HAVING condition;

-- Example: Find subjects with an average marks greater than 85

SELECT subject, AVG(marks) AS average\_marks

FROM Student\_Academic\_Info

GROUP BY subject

**HAVING** AVG(marks) > 85;

-- Example: Find subjects with more than two students who scored above 90

SELECT subject, COUNT(\*) AS total\_students\_above\_90

FROM Student\_Academic\_Info

WHERE marks > 90

GROUP BY subject

**HAVING** COUNT(\*) > 2;

# The **ORDER BY** clause in SQL is used to sort the result set of a query in ascending or descending order based on one or more columns.

SELECT column1, column2, ...

FROM table

ORDER BY column1 [ASC | DESC], column2 [ASC | DESC], ...;

**ASC: Ascending order (default).**

**DESC: Descending order.**

-- Example: Retrieve student names in ascending order based on last names

SELECT first\_name, last\_name

FROM Student\_Personal\_Info

ORDER BY last\_name DESC;

-- Example: Retrieve student names in ascending order based on last names and first names

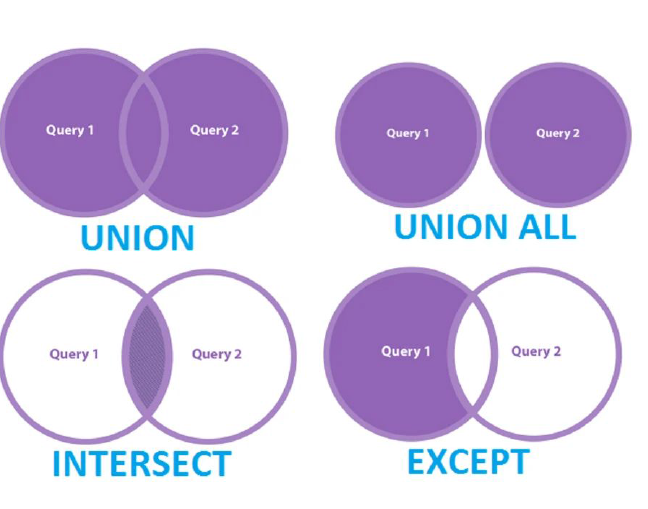
SELECT first\_name, last\_name

FROM Student\_Personal\_Info

ORDER BY last\_name ASC, first\_name ASC;

# Relational set operator

examples using the "Student\_Personal\_Info" and "Student\_Academic\_Info" tables to illustrate each set operator: **UNION, INTERSECT, and EXCEPT.**



-- Example: Combine unique student records from both tables with **UNION**

SELECT student\_id, first\_name, last\_name FROM Student\_Personal\_Info

UNION

SELECT student\_id, first\_name, last\_name FROM Student\_Academic\_Info;

**INTERSECT:**

Retrieve the common student records that exist in both tables.

-- Example: Retrieve common student records between both tables with INTERSECT

SELECT student\_id, first\_name, last\_name FROM Student\_Personal\_Info

INTERSECT

SELECT student\_id, first\_name, last\_name FROM Student\_Academic\_Info;

**EXCEPT:**

Retrieve student records unique to the "Student\_Personal\_Info" table.

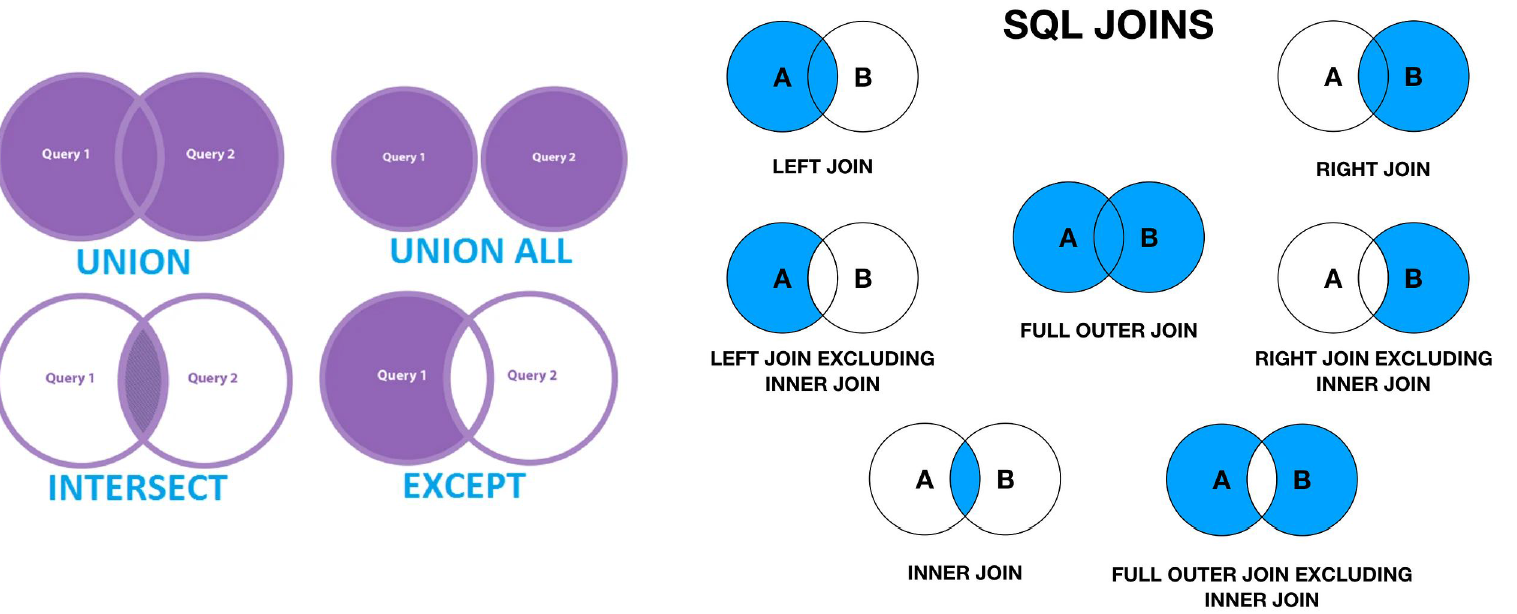
-- Example: Retrieve student records unique to the first table with EXCEPT

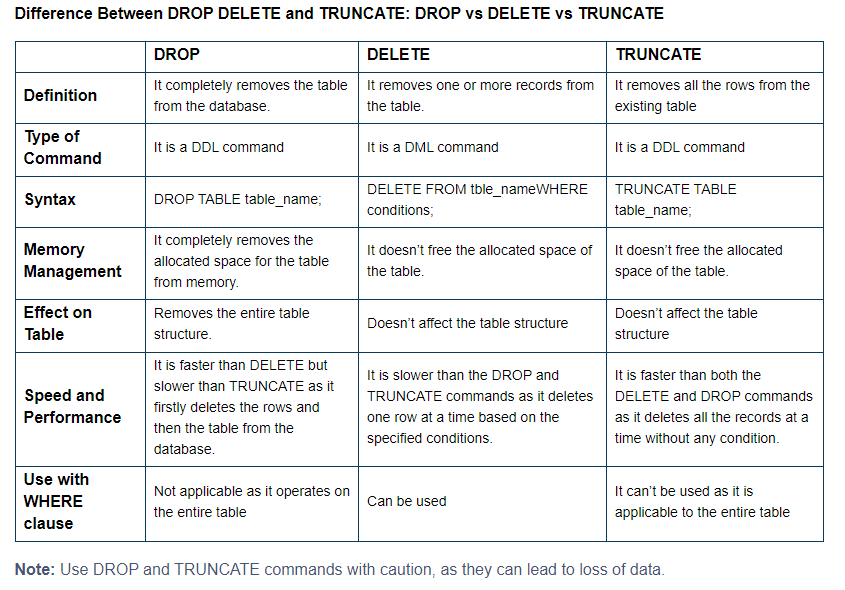
SELECT student\_id, first\_name, last\_name FROM Student\_Personal\_Info

EXCEPT

SELECT student\_id, first\_name, last\_name FROM Student\_Academic\_Info;

**In MySQL, the UNION, INTERSECT, and EXCEPT operators are handled using UNION, INTERSECT, and MINUS, respectively. However, it's important to note that MySQL doesn't directly support INTERSECT and EXCEPT like some other databases do. You can achieve similar results using different approaches.**





**Demonstration**

-- Drop existing tables if they exist

DROP TABLE IF EXISTS Student\_Personal\_Info, Student\_Academic\_Info;

-- Create Student\_Personal\_Info table with additional columns including 'name'

CREATE TABLE Student\_Personal\_Info (

student\_id INT PRIMARY KEY,

first\_name VARCHAR(50),

last\_name VARCHAR(50),

full\_name VARCHAR(100),

date\_of\_birth DATE,

address VARCHAR(100),

email VARCHAR(50)

);

-- Insert sample values into Student\_Personal\_Info

INSERT INTO Student\_Personal\_Info (student\_id, first\_name, last\_name, full\_name, date\_of\_birth, address, email)

VALUES

(1, 'Aarav', 'Sharma', 'Aarav Sharma', '1998-05-15', '123 Main St, City', 'aarav.sharma@example.com'),

(2, 'Ishita', 'Patel', 'Ishita Patel', '1999-08-21', '456 Second St, Town', 'ishita.patel@example.com'),

(3, 'Arjun', 'Verma', 'Arjun Verma', '1997-02-10', '789 Third St, Village', 'arjun.verma@example.com'),

(4, 'Riya', 'Singh', 'Riya Singh', '1996-11-30', '101 Fourth St, Hamlet', 'riya.singh@example.com'),

(5, 'Vedansh', 'Gupta', 'Vedansh Gupta', '1998-09-25', '202 Fifth St, Suburb', 'vedansh.gupta@example.com');

-- Create Student\_Academic\_Info table with additional columns including 'name'

CREATE TABLE Student\_Academic\_Info (

student\_id INT PRIMARY KEY,

full\_name VARCHAR(100),

subject VARCHAR(50),

marks INT,

grade VARCHAR(2)

);

-- Insert sample values into Student\_Academic\_Info

INSERT INTO Student\_Academic\_Info (student\_id, full\_name, subject, marks, grade)

VALUES

(1, 'Aarav Sharma', 'Math', 85, 'A'),

(2, 'Ishita Patel', 'History', 90, 'B+'),

(3, 'Arjun Verma', 'Science', 78, 'B'),

(4, 'Riya Singh', 'English', 92, 'A+'),

(5, 'Vedansh Gupta', 'Math', 88, 'A');

# VIEWs

**7.7 VIRTUAL TABLES: CREATING A VIEW (open textbook 2)**

CREATE VIEW student\_info

as

select student\_personal\_info.student\_id, student\_personal\_info.first\_name, student\_personal\_info.last\_name, student\_personal\_info.birthdate, student\_personal\_info.email, student\_academic\_info.subject, student\_academic\_info.marks, student\_academic\_info.grade

from student\_personal\_info

JOIN student\_academic\_info

on

student\_personal\_info.student\_id=student\_academic\_info.student\_id;

INSERT INTO student\_personal\_info (student\_id, first\_name, last\_name, birthdate, email)

VALUES (6, 'John', 'Doe', '2000-05-20', 'john.doe@example.com');

INSERT INTO student\_academic\_info (student\_id, subject, marks, grade)

VALUES (6, 'Math', 90, 'A');

INSERT INTO student\_academic\_info (student\_id, subject, marks, grade)

VALUES (6, 'Math', 90, 'A');

# Stored procedure

5.2 Functions and Procedures (textbook 1)

**DELIMITER //**

**CREATE PROCEDURE GetStudentDetails (IN studentID INT)**

**BEGIN**

**SELECT \***

**FROM student\_personal\_info**

**WHERE student\_id = studentID;**

**SELECT \***

**FROM student\_academic\_info**

**WHERE student\_id = studentID;**

**END //**

**DELIMITER ;**

Explanation:

* We define a procedure named "GetStudentDetails" that takes one input parameter **studentID**.
* Inside the procedure, we perform two **SELECT** statements to retrieve the student details from the "student\_personal\_info" and "student\_academic\_info" tables based on the provided **studentID**.
* We use the **DELIMITER** command to change the delimiter temporarily to allow the use of semicolons within the stored procedure definition.
* Finally, we change the delimiter back to the default (**;**).

**CALL GetStudentDetails(1);**