### *Exercise 1: List All Books and Their Authors :*

### **Exercise:** Show the name of each author together with the title of the book they wrote and the year in which that book was published.

**Solution :**

SELECT author.Name,book.Title,book.publish\_year FROM author INNER JOIN book ON book.id=author.id;

**Output :**

| Name | Title | Publish\_year |
| --- | --- | --- |
| Marcella Cole | Soulless girl | 2008 |
| Lisa Mullins | Weak Hear | 1980 |
| Dennis Stokes | Faith Of Ligh | 1995 |
| Randolph Vasquez | Memory Of Hope | 2000 |
| Daniel Branson | Warrior Of Wind | 2005 |

*Exercise 2: List Authors and Books Published After 2005 :*

***Exercise:*** *Show the name of each author together with the title of the book they wrote and the year in which that book was published. Show only books published after 2005.*

***Solution:***

*SELECT author.Name,book.Title,book.publish\_year FROM author INNER JOIN book ON book.id=author.id WHERE publish\_year >"2005";*

***Output :***

| Name | Title | Publish\_year |
| --- | --- | --- |
| Marcella Cole | Soulless girl | 2008 |

### E*xercise 3: Show Books Adapted Within 4 Years and Rated Lower Than the Adaptation ;*

***Exercise:*** *For each book, show its title, adaptation title, adaptation year, and publication year.*

*Include only books with a rating lower than the rating of their corresponding adaptation. Additionally, show only those books for which an adaptation was released within four years of the book’s publication.*

*Rename the title column from the book table to book\_title and the title column from the adaptation table to adaptation\_title.*

***Solution:***

*SELECT book.Title AS book\_Title, adaptation.title AS adaptation\_title, book.publish\_year, adaptation.release\_year FROM book JOIN adaptation ON book.id = adaptation.book\_id WHERE adaptation.release\_year -book.publish\_year <= 4 AND book.rating < adaptation.rating;*

***Output:***

| book\_Title | adaptation \_Title | publish\_year | release\_year |
| --- | --- | --- | --- |
| Memory Of Hope | Patrons And Bearers | 2000 | 2004 |

## *LEFT JOIN :*

*Now that you get the gist of INNER JOIN, let’s move on to LEFT JOIN. It’s a type of* [*outer join*](https://learnsql.com/blog/what-is-outer-join/) *that returns all the columns from the left (the first) table and only the matching rows from the right (the second) table. If there is non-matching data, it’s shown as NULL.*

*You can learn more in our* [*article about LEFT JOIN*](https://learnsql.com/blog/what-is-left-join-sql/)*.*

#### **Exercise 4: Show All Books and Their Adaptations (If Any) :**

***Exercise:*** *Show the title of each book together with the title of its adaptation and the date of the release. Show all books, regardless of whether they had adaptations.*

***Solution:***

*SELECT book.title, adaptation.title,adaptation.release\_year FROM book LEFT JOIN adaptation ON book.id = adaptation.book\_id;*

***Output:***

| Title | Title | Release\_year |
| --- | --- | --- |
| Soulless girl | Gone With The Wolves: The Beginning | 2008 |
| Weak Hear | Blacksmith With Silver | 2014 |
| Faith Of Ligh | Companions Of Tomorrow | 2001 |
| Memory Of Hope | Patrons And Bearers | 2004 |
| Warrior Of Wind | Homeless Warrior | 2008 |

### *Exercise 5: Show All Books and Their Movie Adaptations :*

***Exercise:*** *Show all books with their movie adaptations. Select each book's title, the name of its publishing house, the title of its adaptation, and the type of the adaptation. Keep the books with no adaptations in the result.*

***Solution:***

*SELECT book.title,publishing\_house,adaptation.title,adaptation.type FROM book LEFT JOIN adaptation ON book.id = adaptation.book\_id WHERE type = 'movie' OR type IS NULL;*

***Output:***

| Title | Publishing\_house | Title | Type |
| --- | --- | --- | --- |
| Soulless girl | Golden Albatros | Gone With The Wolves: The Beginning | movie |
| Weak Hear | Diarmud Inc | Blacksmith With Silver | movie |
| Faith Of Ligh | White Cloud Press | Companions Of Tomorrow | movie |
| Memory Of Hope | Rutis Enterprises | Patrons And Bearers | movie |
| Warrior Of Wind | Maverick | Homeless Warrior | movie |

#### ***MAIN SQL QUERY :***

CREATE DATABASE Dataset ;

USE Dataset ;

CREATE TABLE author (Id INT PRIMARY KEY,Name CHAR(100),birth\_Year INT ,death\_year INT NULL);

INSERT INTO author VALUES

(1, "Marcella Cole", 1983,NULL),

(2, "Lisa Mullins", 1891, 1950),

(3, "Dennis Stokes", 1935, 1994),

(4, "Randolph Vasquez", 1957, 2004),

(5, "Daniel Branson", 1965, 1990);

CREATE TABLE book (Id INT, author\_id INT NULL, Title CHAR(100), publish\_year INT,publishing\_house CHAR(100),rating FLOAT);

INSERT INTO book VALUES

(1,NULL,"Soulless girl",2008,"Golden Albatros",4.3),

(2,NULL,"Weak Hear",1980,"Diarmud Inc",3.8),

(3,4,"Faith Of Ligh",1995,"White Cloud Press",4.3),

(4,NULL,"Memory Of Hope",2000,"Rutis Enterprises",2.7),

(5,6,"Warrior Of Wind",2005,"Maverick",4.6);

CREATE TABLE adaptation (book\_id INT,type CHAR(50),title CHAR(100),release\_year INT,rating FLOAT);

INSERT INTO adaptation VALUES

(1,"movie","Gone With The Wolves: The Beginning",2008,3),

(3,"movie","Companions Of Tomorrow",2001,4.2),

(5,"movie","Homeless Warrior",2008,4),

(2,"movie","Blacksmith With Silver",2014,4.3),

(4,"movie","Patrons And Bearers",2004,3.2);

CREATE TABLE book\_review (book\_id INT,review CHAR(200),author CHAR (100));

INSERT INTO book\_review VALUES

(1,"An incredible book", "Sylvia Jones"),

(1,"Great, although it has some flaws", "Jessica Parker"),

(2,"Dennis Stokes takes the reader for a ride full of emotions", "Thomas Green"),

(3,"Incredible craftsmanship of the author", "Martin Freeman"),

(4,"Not the best book by this author", "Jude Falth"),

(5,"Claudia Johnson at her best!", "Joe Marqiz"),

(6,"I cannot recall more captivating plot", "Alexander Durham");

SELECT \*FROM author;

SELECT \*FROM book;

SELECT \* FROM adaptation;

SELECT \* FROM book\_review;

### **-- INNER JOIN--**

**Ex -1 :**

SELECT author.Name,book.Title,book.publish\_year FROM author INNER JOIN book ON book.id=author.id;

**Ex -2 :**

SELECT author.Name,book.Title,book.publish\_year FROM author INNER JOIN book ON book.id=author.id WHERE publish\_year >"2005";

**Ex -3 :**

SELECT book.Title AS book\_Title, adaptation.title AS adaptation\_title, book.publish\_year, adaptation.release\_year FROM book JOIN adaptation ON book.id = adaptation.book\_id

WHERE adaptation.release\_year -book.publish\_year <= 4 AND book.rating < adaptation.rating;

**-- LEFT JOIN–**

### **Ex -4:**

SELECT book.title, adaptation.title,adaptation.release\_year FROM book LEFT JOIN adaptatioN ON book.id = adaptation.book\_id;

**Ex -5:**

SELECT book.title,publishing\_house,adaptation.title,adaptation.type FROM book LEFT JOIN adaptation ON book.id = adaptation.book\_id WHERE type = 'movie' OR type IS NULL;

========================= \*\*\*\*\* =========================

## ***FULL JOIN :***

*Here’s another join type that’s useful in some scenarios: the FULL JOIN . This is a LEFT JOIN and RIGHT JOIN put together. It shows matching rows from both tables, rows that have no match from the left table, and rows that have no match from the right table. In short, it shows all data from both tables.*

*You can read more about* [*how and when to use FULL JOIN*](https://learnsql.com/blog/what-is-sql-full-join/)*.*

#### Exercise 6: List All the Books and All the Authors :

**Exercise:** Display the title of each book along with the name of its author. Show all books, even those without an author. Show all authors, even those who haven't published a book yet. Use a FULL JOIN.

**Solution:**

*SELECT title, name FROM book CROSS JOIN author ON book.author\_id = author.id;*

***Output:***

| title | name |
| --- | --- |
| Faith Of Ligh | Randolph Vasquez |

# 

## SQL PRACTICE - 2 (DATASET-3)

# **MAIN SQL QUERY :**

# CREATE DATABASE Dataset3;

# USE Dataset3;

CREATE TABLE workshop\_workers (id INT, name VARCHAR(100), specialization VARCHAR(150),master\_id INT NULL, experience INT, project\_id INT);

INSERT INTO workshop\_workers VALUES

(1,"Mathew Conn","woodworking",NULL,20,1),

(2,"Kate Brown","woodworking",1,4,1),

(3,"John Doe","incrusting",5,3,1),

(4,"John Kowalsky","watchmaking",7,2,3),

(5,"Suzan Gregowitch","incrusting",NULL,15,4);

SELECT \* FROM workshop\_workers;

]***Output:***

| id | name | specialization | master\_id | experience | project\_id |
| --- | --- | --- | --- | --- | --- |
| 1 | Mathew Conn | woodworking |  | 20 | 1 |
| 2 | Kate Brown | woodworking | 1 | 4 | 1 |
| 3 | John Doe | incrusting | 5 | 3 | 1 |
| 4 | John Kowalsky | watchmaking | 7 | 2 | 3 |
| 5 | Suzan Gregowitch | incrusting |  | 15 | 4 |

# 

# *Exercise 1: List All Workers and Their Direct Supervisors :*

***Exercise:*** *Show all workers' names together with the names of their direct supervisors. Rename the columns apprentice\_name and master\_name, respectively. Consider only workers who have a supervisor (i.e. a master).*

***Solution:***

SELECT apprentice.name AS apprentice\_name, master.name AS master\_name FROM workshop\_workers apprentice JOIN workshop\_workers master ON apprentice.master\_id = master.id;

***Output:***

| apprentice\_name | master\_name |
| --- | --- |
| Kate Brown | Mathew Conn |
| John Doe | Suzan Gregowitch |

**EX -1 :**

SELECT apprentice.name AS apprentice\_name, master.name AS master\_name FROM workshop\_workers apprentice JOIN workshop\_workers master ON apprentice.master\_id = master.id;

***========================== \*\*\*\* ========================***

## **TASK CASES - 5**

##### **### Case Study 1:** List All Books and Their Authors :

\*\*Scenario\*\*: You have two tables: `books` and `authors`. You need to list all books along with their authors.

**Solution :**

SELECT book.Title,author.Name FROM book INNER JOIN author ON book.Id= author.Id;

***Output:***

| Title | Name |
| --- | --- |
| Soulless girl | Marcella Cole |
| Weak Hear | Lisa Mullins |
| Faith Of Ligh | Dennis Stokes |
| Memory Of Hope | Randolph Vasquez |
| Warrior Of Wind | Daniel Branson |

##### **### Case Study 2:** List Authors and Books Published After 2005 :

*\*\*Scenario\*\*: You need to list authors and the titles of books published after 2005.*

***Solution:***

SELECT author.Name,book.Title FROM author INNER JOIN book ON book.id=author.id WHERE publish\_year >"2005";

***Output:***

| *Name* | *Title* |
| --- | --- |
| *Marcella Cole* | *Soulless girl* |

##### **### Case Study 3:** Show Books Adapted Within 4 Years and Rated Lower Than the Adaptation :

*\*\*Scenario\*\*: You have tables `books` and `adaptations`. You need to show books that were adapted within 4 years and rated lower than the adaptation.*

***Solution:***

SELECT book.Title AS book\_Title, adaptation.title AS adaptation\_title, book.rating, adaptation.rating AS adaptation\_rating FROM book JOIN adaptation ON book.id = adaptation.book\_id WHERE adaptation.release\_year -book.publish\_year <= 4 AND book.rating < adaptation.rating;

***Output:***

| Book\_Title | Adaptation\_Title | Rating | Adaptation\_Rating |
| --- | --- | --- | --- |
| Memory Of Hope | Patrons And Bearers | 2.7 | 3.2 |

##### ### Case Study 4: Show All Books and Their Movie Adaptations (If Any) :

*\*\*Scenario\*\*: You have tables `books` and `movies`. You need to show all books and their movie adaptations, if any*.

**Solution:**

SELECT book.title AS book\_Title,adaptation.title AS adaptation\_Title FROM book LEFT JOIN adaptation ON book.id = adaptation.book\_id WHERE type = 'movie' OR type IS NULL;

***Output:***

| Book\_Title | Adaptation\_Title |
| --- | --- |
| Soulless girl | Gone With The Wolves: The Beginning |
| Weak Hear | Blacksmith With Silver |
| Faith Of Ligh | Companions Of Tomorrow |
| Memory Of Hope | Patrons And Bearers |
| Warrior Of Wind | Homeless Warrior |

##### **### Case Study 5:** List All Products, Prices, Producers, and Departments :

*\*\*Scenario\*\*: You have tables `products`, `producers`, and `departments`. You need to list all products along with their prices, producers, and departments.*

**Solution:**

SELECT \* FROM case5;

***Output:***

| Name | Price | Producer\_Name | Department\_Name |
| --- | --- | --- | --- |
| Product 1 | 10.99 | Producer A | Department 1 |
| Product 2 | 15.99 | Producer B | Department 2 |
| Product 3 | 12.99 | Producer C | Department 3 |

## **CONSTRAINTS :**

### ***TASK 1: Age Constraint :***

*Scenario : Ensure that the `age` column only accepts values between 18 and 65.*

*CREATE TABLE task1( name VARCHAR(100), age INT, CHECK (age >=18- 65), gender VARCHAR(50) NOT NULL, salary INT, city VARCHAR(100) ,PAN VARCHAR(100) UNIQUE);*

INSERT INTO task1 (name, age, gender, salary,city, PAN)

VALUES ('John Doe', 70, 'Male', 20000,’Mumbai’, 'ABCDE1234F');

**Error :**

13:41:05 INSERT INTO task1 (name, age, gender, salary, city, PAN) VALUES ('John Doe', 70, 'Male', 20000, 'Mumbai', 'ABCDE1234F') Error Code: 1146. Table 'cons.mydata' doesn't exist 0.000 sec,

***Error Solved :***

*CREATE TABLE task1( name VARCHAR(100), age INT, CHECK (age >=18- 65), gender VARCHAR(50) NOT NULL, salary INT, city VARCHAR(100) ,PAN VARCHAR(100) UNIQUE);*

INSERT INTO task1 (name, age, gender, salary,city, PAN)

VALUES ('John Doe', 35, 'Male', 20000, 'mumbai', 'ABCDE1234F');

SELECT \* FROM task1;

***Output:***

| Name | Age | Gender | Salary | City | PAN |
| --- | --- | --- | --- | --- | --- |
| John Doe | 35 | Male | 20000 | Mumbai | ABCDE1234F |

### ***TASK 2: Gender Constraint :***

Scenario : Ensure that the `gender` column only accepts values 'Male', 'Female', or 'Other'.

*CREATE TABLE task2 (name VARCHAR(100), age INT, gender VARCHAR(50) NOT NULL, CHECK(gender = "male"), salary INT, city VARCHAR(100), PAN VARCHAR(100) UNIQUE);*

*INSERT INTO task2 (name, age, gender, salary, city, PAN)*

*VALUES ('Jane Doe', 25, 'Unknown', 25000,’Delhi’, 'FGHIJ5678K');*

**Error :**

09:56:54 INSERT INTO task2 (name, age, gender, salary, city, PAN) VALUES ('Jane Doe', 25, 'Unknown', 25000,’Delhi’, 'FGHIJ5678K') Error Code: 3819. Check constraint 'task1\_chk\_1' is violated. 0.000 sec,

***Error Solved :***

CREATE TABLE task2 (name VARCHAR(100), age INT, gender VARCHAR(50) NOT NULL, CHECK(gender = "male"), salary INT, city VARCHAR(100) ,PAN VARCHAR(100) UNIQUE);

INSERT INTO task2 (name, age, gender, salary, city, PAN)

VALUES ('Jane Doe', 25, 'Male', 25000, 'Delhi', 'FGHIJ5678K');

SELECT \* FROM task2 ;

***Output:***

| Name | Age | Gender | Salary | City | PAN |
| --- | --- | --- | --- | --- | --- |
| John Doe | 35 | Male | 20000 | Delhi | ABCDE1234F |

### ***TASK 3: Salary Constraint :***

Scenario : Ensure that the `salary` column only accepts values greater than 10000 (already

present).

*CREATE TABLE task3 (name VARCHAR(100), age INT, gender VARCHAR(50) NOT NULL, salary INT, CHECK (salary<=10000), city VARCHAR(100), PAN VARCHAR(100) UNIQUE);*

*INSERT INTO task3 (name, age, gender, salary, city, Pan)*

*VALUES ('Alice Brown', 30, 'Female', 90000, 'Chennai', 'KLMNO12345');*

**Error :**

*10:32:21 INSERT INTO task3 (name, age, gender, salary, city, Pan) VALUES ('Alice Brown', 30, 'Female', 90000, 'Chennai', 'KLMNO12345') Error Code: 3819. Check constraint 'task1\_chk\_1' is violated. 0.000 sec*

***Error Solved :***

*CREATE TABLE task3 (name VARCHAR(100), age INT, gender VARCHAR(50) NOT NULL, salary INT, CHECK (salary<=10000), city VARCHAR(100), PAN VARCHAR(100) UNIQUE);*

INSERT INTO task3 (name, age, gender, salary, city, Pan)

VALUES ('Alice Brown', 30, 'Female', 9000, 'Chennai', 'KLMNO12345');

SELECT \* FROM task3 ;

***Output:***

| *Name* | *Age* | *Gender* | *Salary* | *City* | *PAN* |
| --- | --- | --- | --- | --- | --- |
| Alice Brown | 30 | Female | 9000 | Chennai | KLMNO12345 |

#### 

### ***TASK 4: PAN Unique Constraint :***

Scenario : Ensure that the `Pan` column only accepts unique values.

*CREATE TABLE task4 (name VARCHAR(100), age INT, gender VARCHAR(50) NOT NULL, salary INT, city VARCHAR(100), PAN VARCHAR(100) UNIQUE);*

*INSERT INTO task4 (name, age, gender, salary, city, PAN) VALUES*

*('John Doe', 35, 'Male', 20000, 'Mumbai', 'ABCDE1234F'),*

*('Robert Smith', 40, 'Male', 30000, 'Hyderabad', 'ABCDE1234F');*

**Error :**

10:51:58 INSERT INTO task4 (name, age, gender, salary, city, PAN) VALUES

('John Doe', 35, 'Male', 20000, 'Mumbai', 'ABCDE1234F'), ('Robert Smith', 40, 'Male', 30000, 'Hyderabad', 'ABCDE1234F') Error Code: 1062. Duplicate entry 'ABCDE1234F' for key 'task1.PAN' 0.000 sec

***Error Solved :***

CREATE TABLE task4 (name VARCHAR(100), age INT, gender VARCHAR(50) NOT NULL, salary INT, city VARCHAR(100), PAN VARCHAR(100) UNIQUE);

INSERT INTO task4 (name, age, gender, salary, city, PAN) VALUES

('John Doe', 35, 'Male', 20000, 'Mumbai', 'ABCDE1234F'),

('Robert Smith', 40, 'Male', 30000, 'Hyderabad', 'ABCDF1234F');

SELECT \* FROM task4 ;

***Output:***

| Name | Age | Gender | Salary | City | PAN |
| --- | --- | --- | --- | --- | --- |
| John Doe | 35 | Male | 20000 | Mumbai | ABCDE1234F |
| Robert Smith | 40 | Male | 30000 | Hyderabad | ABCDF1234F |

### ***TASK 5: City Default Value :***

Scenario : Ensure that the `city` column defaults to 'Chennai' if not provided.

*CREATE TABLE task5 (name VARCHAR(100), age INT, gender VARCHAR(50) NOT NULL, salary INT, city VARCHAR(100) DEFAULT 'Chennai', PAN VARCHAR(100) UNIQUE);*

*INSERT INTO task5 (name, age, gender, salary, Pan)*

*VALUES ('Mary Johnson', 28, 'Female', 15000, 'ZYXWV09876');*

SELECT \* FROM task5 ;

***Output:***

| Name | Age | Gender | Salary | City | PAN |
| --- | --- | --- | --- | --- | --- |
| Mary Johnson | 28 | Female | 15000 | Chennai | ZYXWV09876 |

### ***TASK 6: Email Unique Constraint :***

Scenario: Ensure that the `email` column only accepts unique values.

*CREATE TABLE task6 (name VARCHAR(100), age INT, gender VARCHAR(50) NOT NULL, salary INT, city VARCHAR(100) DEFAULT 'Chennai', PAN VARCHAR(100), Email VARCHAR(100) UNIQUE);*

*INSERT INTO task6 (name, age, gender, salary, city, Pan, email) VALUES*

*('John Doe', 30, 'Male', 25000, 'Bangalore', 'ABCDE1234F', 'john@example.com'),*

*('Jane Smith', 32, 'Female', 27000, 'Mumbai', 'FGHIJ5678K', 'john@example.com');*

**Error :**

11:17:10 INSERT INTO task6 (name, age, gender, salary, city, Pan, email)

VALUES ('John Doe', 30, 'Male', 25000, 'Bangalore', 'ABCDE1234F', '[john@example.com](mailto:john@example.com)'), ('Jane Smith', 32, 'Female', 27000, 'Mumbai', 'FGHIJ5678K', '[john@example.com](mailto:john@example.com)') Error Code: 1062. Duplicate entry 'john@example.com' for key 'task1.Email' 0.000 sec

***Error Solved :***

*CREATE TABLE task6 (name VARCHAR(100), age INT, gender VARCHAR(50) NOT NULL, salary INT, city VARCHAR(100) DEFAULT 'Chennai', PAN VARCHAR(100), Email VARCHAR(100) UNIQUE);*

*INSERT INTO task6 (name, age, gender, salary, city, Pan, email) VALUES*

*('John Doe', 30, 'Male', 25000, 'Bangalore', 'ABCDE1234F', '*[*john@example.com*](mailto:john@example.com)*'),*

*('Jane Smith', 32, 'Female', 27000, 'Mumbai', 'FGHIJ5678K', '*[*jane@example.com*](mailto:jane@example.com)*');*

SELECT \* FROM task6 ;

***Output:***

| Name | Age | Gender | Salary | City | PAN | Email |
| --- | --- | --- | --- | --- | --- | --- |
| John Doe | 30 | Male | 25000 | Bangalore | ABCDE1234F | john@example.com |
| Jane Smith | 32 | Female | 27000 | Mumbai | FGHIJ5678K | jane@example.com |

### ***TASK 7: Not Null Constraint on Fullname :***

Scenario: Ensure that the `Fullname` column does not accept null values.

CREATE TABLE task7 (name VARCHAR(100) NOT NULL, age INT, gender VARCHAR(50) NOT NULL, salary INT, city VARCHAR(100), PAN VARCHAR(50));

INSERT INTO task7 (age, gender, salary, city, Pan)

VALUES (25, 'Male', 20000, 'Chennai', 'KLMNO12345');

**Error :**

11:36:53 INSERT INTO task7 (age, gender, salary, city, Pan) VALUES (25, 'Male', 20000, 'Chennai', 'KLMNO12345') Error Code: 1364. Field 'name' doesn't have a default value 0.000 sec

***Error Solved :***

CREATE TABLE task7 (name VARCHAR(100) NOT NULL, age INT, gender VARCHAR(50) NOT NULL, salary INT, city VARCHAR(100), PAN VARCHAR(50));

INSERT INTO task7 (name,age, gender, salary, city, Pan)

VALUES ('John',25, 'Male', 20000, 'Chennai', 'KLMNO12345');

SELECT \* FROM task7 ;

***Output:***

| Name | Age | Gender | Salary | City | PAN |
| --- | --- | --- | --- | --- | --- |
| John | 25 | Male | 20000 | Chennai | KLMNO12345 |

### ***TASK 8: Default Value for Gender :***

Scenario: Ensure that the `gender` column defaults to 'Not Specified' if not provided.

CREATE TABLE task8 (name VARCHAR(100), age INT, gender VARCHAR(50) DEFAULT 'Female',salary INT, city VARCHAR(100) ,PAN VARCHAR(100) UNIQUE);

INSERT INTO task8 (name, age, salary, city, Pan)

VALUES ('Alice Brown', 28, 22000, 'Chennai', 'LMNOP67890');

SELECT \* FROM task8 ;

***Output:***

| Name | Age | Gender | Salary | City | PAN |
| --- | --- | --- | --- | --- | --- |
| Alice Brown | 28 | Female | 22000 | Chennai | LMNOP67890 |

### 

### ***TASK 9: Mobile Number Constraint :***

Scenario: Ensure that the `mobile` column only accepts 10-digit numbers.

CREATE TABLE task9 (name VARCHAR(100), age INT, gender VARCHAR(50) , salary INT, city VARCHAR(100) ,PAN VARCHAR(100) UNIQUE, mobile VARCHAR(15), CHECK ( LENGTH(mobile) = 10);

INSERT INTO task9 (name, age, gender, salary, city, Pan, mobile)

VALUES ('Raj Patel', 35, 'Male', 28000, 'Ahmedabad', 'QRSTU98765', '123456789');

**Error :**

21:13:58 INSERT INTO task9 (name, age, gender, salary, city, Pan, mobile) VALUES ('Raj Patel', 35, 'Male', 28000, 'Ahmedabad', 'QRSTU98765', '123456789') Error Code: 3819. Check constraint 'task9\_chk\_1' is violated. 0.015 sec

***Error Solved :***

CREATE TABLE task9 (name VARCHAR(100), age INT, gender VARCHAR(50) , salary INT, city VARCHAR(100) ,PAN VARCHAR(100) UNIQUE, mobile VARCHAR(15), CHECK (LENGTH(mobile) = 10));

INSERT INTO task9 (name, age, gender, salary, city, PAN, mobile)

VALUES ('Raj Patel', 35, 'Male', 28000, 'Ahmedabad', 'QRSTU98765', '1234567890');

SELECT \* FROM task9;

***Output:***

| Name | Age | Gender | Salary | City | PAN | *Mobile* |
| --- | --- | --- | --- | --- | --- | --- |
| *Raj Patel* | *35* | *Male* | *28000* | *Ahmedabad* | *QRSTU98765* | *1234567890* |

### 

### ***TASK 10: Department Foreign Key Constraint :***

*Scenario: Ensure that the `department\_id` in `mydata` references `id` in `departments` table.*

*CREATE TABLE dep( id INT PRIMARY KEY,name VARCHAR(50));*

*CREATE TABLE task10(name VARCHAR(100) NOT NULL, age INT, gender VARCHAR(50) NOT NULL, salary INT, city VARCHAR(100), PAN VARCHAR(50),dep\_id INT,FOREIGN KEY(dep\_id) REFERENCES dep(id));*

*INSERT INTO task10 (name, age, gender, salary, city, Pan, dep\_id)*

*VALUES ('Ravi Kumar', 29, 'Male', 24000, 'Hyderabad', 'VWXYZ54321', 99);*

**Error :**

10:36:55 INSERT INTO task10 (name, age, gender, salary, city, Pan, dep\_id) VALUES ('Ravi Kumar', 29, 'Male', 24000, 'Hyderabad', 'VWXYZ54321', 99)

Error Code: 1452. Cannot add or update a child row: a foreign key constraint fails (`cons`.`task10`, CONSTRAINT `task10\_ibfk\_1` FOREIGN KEY (`dep\_id`) REFERENCES `dep` (`id`)) 0.016 sec

### ***TASK 11: Positive Age Constraint :***

Scenario: Ensure that the `age` column only accepts positive values.

CREATE TABLE task11( name VARCHAR(100), age INT, CHECK (age >0), gender VARCHAR(50) , salary INT, city VARCHAR(100) ,PAN VARCHAR(100));

INSERT INTO task11 (name, age, gender, salary, city, PAN)

VALUES ('Priya Singh', -25, 'Female', 30000, 'Delhi', 'ABCDE67890');

**Error :**

10:51:30 INSERT INTO task11 (name, age, gender, salary, city, PAN) VALUES ('Priya Singh', -25, 'Female', 30000, 'Delhi', 'ABCDE67890') Error Code: 3819. Check constraint 'task11\_chk\_1' is violated. 0.000 sec

***Error Solved :***

CREATE TABLE task11( name VARCHAR(100), age INT, CHECK (age >0), gender VARCHAR(50) , salary INT, city VARCHAR(100) ,PAN VARCHAR(100));

INSERT INTO task11 (name, age, gender, salary, city, PAN)

VALUES ('Priya Singh', 45, 'Female', 30000, 'Delhi', 'ABCDE67890');

SELECT \* FROM task11;

***Output:***

| Name | Age | Gender | Salary | City | PAN |
| --- | --- | --- | --- | --- | --- |
| Priya Singh | 45 | Female | 30000 | Delhi | ABCDE67890 |

### ***TASK 12: Fullname Length Constraint :***

Scenario: Ensure that the `Fullname` column only accepts names between 3 and 20 characters.

*CREATE TABLE task12( name VARCHAR(100),CHECK(LENGTH(name>=3) AND(LENGTH(name<=20))), age INT, gender VARCHAR(50) , salary INT, city VARCHAR(100) ,PAN VARCHAR(100));*

*INSERT INTO task12(name, age, gender, salary, city, PAN)*

*VALUES ('Jo', 32, 'Male', 25000, 'Chennai', 'PQRS123456');*

**Error :**

11:10:23 INSERT INTO task12(name, age, gender, salary, city, PAN) VALUES ('Jo', 32, 'Male', 25000, 'Chennai', 'PQRS123456') Error Code: 3819. Check constraint 'task12\_chk\_1' is violated. 0.000 sec

***Error Solved :***

CREATE TABLE task12( name VARCHAR(100),CHECK(LENGTH(name>=3) AND(LENGTH(name<=20))), age INT, gender VARCHAR(50) , salary INT, city VARCHAR(100) ,PAN VARCHAR(100));

INSERT INTO task12(name, age, gender, salary, city, PAN)

VALUES ('John', 32, 'Male', 25000, 'Chennai', 'PQRS123456');

SELECT \* FROM task12 ;

***Output:***

| Name | Age | Gender | Salary | City | PAN |
| --- | --- | --- | --- | --- | --- |
| John | 32 | Male | 25000 | Chennai | PQRS123456 |

### ***TASK 13: City Not Null Constraint :***

Scenario: Ensure that the `city` column does not accept null values.

*CREATE TABLE task13( name VARCHAR(100), age INT, gender VARCHAR(50) , salary INT, city VARCHAR(100) NOT NULL ,PAN VARCHAR(100));*

*INSERT INTO task13 (name, age, gender, salary, PAN)*

*VALUES ('Amit Shah', 40, 'Male', 32000, 'WXYZ987654');*

**Error :**

11:20:40 INSERT INTO task13 (name, age, gender, salary, PAN) VALUES ('Amit Shah', 40, 'Male', 32000, 'WXYZ987654') Error Code: 1364. Field 'city' doesn't have a default value 0.000 sec

***Error Solved :***

CREATE TABLE task13( name VARCHAR(100), age INT, gender VARCHAR(50) , salary INT, city VARCHAR(100) NOT NULL ,PAN VARCHAR(100));

INSERT INTO task13 (name, age, gender, salary,city, PAN)

VALUES ('Amit Shah', 40, 'Male', 32000,'Chennai', 'WXYZ987654');

SELECT \* FROM task13 ;

***Output:***

| Name | Age | Gender | Salary | City | PAN |
| --- | --- | --- | --- | --- | --- |
| Amit Shah | 40 | Male | 32000 | Chennai | WXYZ987654 |

### ***TASK 14: Age Default Value :***

Scenario: Ensure that the `age` column defaults to 25 if not provided.

CREATE TABLE task14( name VARCHAR(100), age INT DEFAULT 25, gender VARCHAR(50) , salary INT, city VARCHAR(100) ,PAN VARCHAR(100));

INSERT INTO task14 (name,gender,salary,city,PAN)

VALUES ( "joo","female",9000,"chennai", "ABCDE1234F");

SELECT \* FROM task14;

***Output:***

| Name | Age | Gender | Salary | City | PAN |
| --- | --- | --- | --- | --- | --- |
| joo | 25 | female | 9000 | chennai | ABCDE1234F |

### ***TASK 15: Salary Between Range Constraint :***

Scenario: Ensure that the `salary` column only accepts values between 15000 and 50000.

*CREATE TABLE task15( name VARCHAR(100),age INT, gender VARCHAR(50) ,salary INT,CHECK (salary>=15000 AND salary<=50000),city VARCHAR(100) , PAN VARCHAR(100));*

*INSERT INTO task15 (name, age, gender, salary, city, PAN)*

*VALUES ('Kiran Rao', 45, 'Male', 10000, 'Kolkata', 'HIJKL98765');*

**Error :**

11:42:49 INSERT INTO task15 (name, age, gender, salary, city, PAN) VALUES ('Kiran Rao', 45, 'Male', 10000, 'Kolkata', 'HIJKL98765') Error Code: 3819. Check constraint 'task15\_chk\_1' is violated. 0.000 sec

***Error Solved :***

CREATE TABLE task15( name VARCHAR(100),age INT, gender VARCHAR(50) ,salary INT,CHECK (salary>=15000 AND salary<=50000),city VARCHAR(100) , PAN VARCHAR(100));

INSERT INTO task15 (name, age, gender, salary, city, PAN)

VALUES ('Kiran Rao', 45, 'Male', 35000, 'Kolkata', 'HIJKL98765');

SELECT \* FROM task15;

***Output:***

| Name | Age | Gender | Salary | City | PAN |
| --- | --- | --- | --- | --- | --- |
| Kiran Rao | 45 | Male | 35000 | Kolkata | HIJKL98765 |

### 

### TASK 16: Country Default Value

Scenario: Add a `country` column and ensure it defaults to 'India' if not provided.

CREATE TABLE task16( name VARCHAR(100),age INT, gender VARCHAR(50) ,salary INT,city VARCHAR(100) , PAN VARCHAR(100));

INSERT INTO task16 (name, age, gender, salary, city, PAN)

VALUES ('Rohit Sharma', 35, 'Male', 35000, 'Mumbai', 'MNO12345');

ALTER TABLE task16 ADD COLUMN country VARCHAR(50) DEFAULT 'INDIA’;

SELECT \* FROM task16;

***Output:***

| Name | Age | Gender | Salary | City | PAN | Country |
| --- | --- | --- | --- | --- | --- | --- |
| Rohit Sharma | 35 | Male | 35000 | Mumbai | MNO12345 | INDIA |

## ***Case Study 1 : Employee Management System***

# Scenario:

You have been hired as a database administrator for a new company, and you need to create an Employee Management System to keep track of the employees, their departments, projects they are working on, and their attendance.

CREATE DATABASE Employee\_Management;

USE Employee\_Management;

# .CREATE TABLE department (id INT AUTO\_INCREMENT PRIMARY KEY, dep\_name VARCHAR(50) NOT NULL);

INSERT INTO department (dep\_name)

VALUES

('HR'),

('IT'),

('TESTING'),

('DVELOPER'),

('MANAGEMENT');

SELECT \* FROM department;

# .CREATE TABLE empolyees (emp\_id INT AUTO\_INCREMENT PRIMARY KEY, emp\_name VARCHAR(50) NOT NULL,age INT NOT NULL,gender VARCHAR(10) NOT NULL,salary INT NOT NULL,CHECK (salary>30000),dep\_id INT ,FOREIGN KEY (dep\_id) REFERENCES department (id));

INSERT INTO employees VALUES

('Akhil',25,'Male',55000,3),

('Bavya',20,'Female',68000,5),

('Anbu',35,'Male',59000,2),

('Vinoth',40,'Male',74000,4),

('Pavi',23,'Female',76000,3),

('Yuva',29,'Male',59000,1),

('Divya',25,'Female',67000,2),

('Gomathi',23,'Female',82000,1),

('Manisha',26,'Female',58000,4),

('Deepak',48,'Male',65000,2),

('Vaishu',55,'Female',74000,3),

('Anish',36,'Male',57000,1),

('Vinitha',49,'Female',69000,2),

('Deekshit',55,'Male',79000,5),

('Dharani',45,'Female',85000,2);

SELECT \* FROM empolyees ;

# .CREATE TABLE projects(id INT AUTO\_INCREMENT PRIMARY KEY, name VARCHAR(50) NOT NULL,Budget INT NOT NULL,CHECK (Budget>10000));

INSERT INTO projects VALUES

('App Develop',75000),

('Web Develop',95000),

('Data Analysis',65000),

('Network',80000),

('Game Develop',99000);

SELECT \* FROM projects;

# .CREATE TABLE Emp\_Project (emp\_id INT, project\_id INT, FOREIGN KEY (emp\_id) REFERENCES employees(emp\_id),FOREIGN KEY (project\_id) REFERENCES projects(id) ,PRIMARY KEY ( emp\_id, project\_id ));

INSERT INTO Emp\_Project VALUES

(1,4),

(3,1),

(2,3),

(4,5),

(5,42);

SELECT \* FROM Emp\_Project;

#. CREATE TABLE Attendance (id INT AUTO\_INCREMENT PRIMARY KEY,emp\_id INT ,FOREIGN KEY (emp\_id) REFERENCES employees(emp\_id),Date DATE NOT NULL,Status ENUM ('Present', 'Absent', 'On Leave'));

INSERT INTO Attendance (emp\_id, Date, Status) VALUES

(1,'2024-04-24','Present'),

(2,'2024-04-24','ON Leave'),

(6,'2024-04-24','Absent'),

(8,'2024-04-24','Present'),

(4,'2024-04-24','Present'),

(10,'2024-04-24','Absent'),

(3,'2024-04-24','Present'),

(7,'2024-04-24','ON Leave'),

(12,'2024-04-24','Present'),

(15,'2024-04-24','Absent'),

(5,'2024-04-24','Absent'),

(11,'2024-04-24','Present'),

(14,'2024-04-24','ON Leave'),

(9,'2024-04-24','Present'),

(13,'2024-04-24','Absent');

SELECT \* FROM Attendance;

## **Queries:**

### ***1. Retrieve all employees :***

SELECT \* FROM Employees;

**Output :**

| Emp\_id | Emp\_name | Age | Gender | Salary | Dep\_id |
| --- | --- | --- | --- | --- | --- |
| 1 | Akhil | 25 | Male | 55000 | 3 |
| 2 | Bavya | 20 | Female | 68000 | 5 |
| 3 | Anbu | 35 | Male | 59000 | 2 |
| 4 | Vinoth | 40 | Male | 74000 | 4 |
| 5 | Pavi | 23 | Female | 76000 | 3 |
| 6 | Yuva | 29 | Male | 59000 | 1 |
| 7 | Divya | 25 | Female | 67000 | 2 |
| 8 | Gomathi | 23 | Female | 82000 | 1 |
| 9 | Manisha | 26 | Female | 58000 | 4 |
| 10 | Deepak | 48 | Male | 65000 | 2 |
| 11 | Vaishu | 55 | Female | 74000 | 3 |
| 12 | Anish | 36 | Male | 57000 | 1 |
| 13 | Vinitha | 49 | Female | 69000 | 2 |
| 14 | Deekshit | 55 | Male | 79000 | 5 |
| 15 | Dharani | 45 | Female | 85000 | 2 |

### ***2. Retrieve all employees in the 'IT' department :***

SELECT Employees.emp\_name,department.dep\_name FROM Employees JOIN department ON Employees.dep\_id = department.id WHERE dep\_name = 'IT';

| Emp\_name | Dep\_name |
| --- | --- |
| Anbu | IT |
| Divya | IT |
| Deepak | IT |
| Vinitha | IT |
| Dharani | IT |

### ***3. List all projects with a budget greater than 50000 :***

SELECT name AS Pro\_Name,budget FROM Projects WHERE budget > 50000;

**Output :**

| Project\_name | Budget |
| --- | --- |
| App Develop | 75000 |
| Web Develop | 95000 |
| Data Analysis | 65000 |
| Network | 80000 |
| Game Develop | 99000 |

### ***4. Show the names of employees and the projects they are working on. :***

SELECT Employees.emp\_name,projects.name FROM Employees JOIN projects ON Employees.emp\_id = projects.id;

**Output :**

| *Emp\_name* | *Project\_name* |
| --- | --- |
| *Akhil* | *App Develop* |
| *Bavya* | *Web Develop* |
| *Anbu* | *Data Analysis* |
| *Vinoth* | *Network* |
| *Pavi* | *Game Develop* |

### ***5. Count the number of employees in each department :***

SELECT COUNT( Employees.emp\_id),department.dep\_name FROM Employees JOIN department ON department.id = Employees.emp\_id GROUP BY department.id***;***

**Output :**

| count\_emp\_id | dep\_name |
| --- | --- |
| 1 | HR |
| 1 | IT |
| 1 | TESTING |
| 1 | DEVELOPER |
| 1 | MANAGEMENT |

### 

### ***6. Find employees with a salary greater than 50000.***

SELECT Employees.emp\_name,Employees.salary FROM Employees WHERE salary>50000;

**Output :**

| emp\_name | salary |
| --- | --- |
| Akhil | 55000 |
| Bavya | 68000 |
| Anbu | 59000 |
| Vinoth | 74000 |
| Pavi | 76000 |
| Yuva | 59000 |
| Divya | 67000 |
| Gomathi | 82000 |
| Manisha | 58000 |
| Deepak | 65000 |
| Vaishu | 74000 |
| Anish | 57000 |
| Vinitha | 69000 |
| Deekshit | 79000 |
| Dharani | 85000 |

### ***7.Get attendance records for a specific employee :***

SELECT Attendance.Date,Attendance.Status FROM Attendance WHERE Attendance.emp\_id=7;

**Output :**

| Date | Status |
| --- | --- |
| 2024-04-24 | On Leave |

## **## Update Records:**

### ***1. Increase the salary of employees in the 'HR' department by 10% :***

UPDATE Employees JOIN department ON Employees.dep\_id=department.id SET Employees.salary= Employees.salary \*1.10 WHERE department.dep\_name='HR';

SELECT Employees.emp\_name,Employees.salary FROM Employees JOIN department ON Employees.dep\_id = department.id WHERE department.dep\_name='HR';

**Output :**

| emp\_name | salary |
| --- | --- |
| Yuva | 64900 |
| Gomathi | 90200 |
| Anish | 62700 |

### 

### ***2. Change the department of an employee :***

UPDATE Employees SET dep\_id = (SELECT id FROM department WHERE dep\_name = 'TESTING' LIMIT 1) WHERE dep\_id = 2 ;

SELECT Employees.emp\_name, department.dep\_name FROM Employees JOIN department ON Employees.dep\_id = department.id WHERE department.dep\_name = 'TESTING';

**Output :**

| emp\_name | dep\_name |
| --- | --- |
| Akhil | TESTING |
| Anbu | TESTING |
| Pavi | TESTING |
| Divya | TESTING |
| Deepak | TESTING |
| Vaishu | TESTING |
| Vinitha | TESTING |
| Dharani | TESTING |

## ## Delete Records:

### ***1.Delete a project that is completed :***

DELETE FROM Projects WHERE name = 'NETWORK' ;

**Output:**

### **?**

### 

### ***2.Remove an employee who has resigned :***

**?**

**## Constraints and Modifications:**

### ***1.Add a unique constraint to the `email` column in `Employees` table :***

ALTER TABLE Employees ADD COLUMN email varchar(50) UNIQUE;

SELECT \* FROM Employees ;

**Output :**

| emp\_id | emp\_name | age | gender | salary | dep\_id | email |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | Akhil | 25 | Male | 55000 | 3 | NULL |
| 2 | Bavya | 20 | Female | 68000 | 5 | NULL |
| 3 | Anbu | 35 | Male | 59000 | 3 | NULL |
| 4 | Vinoth | 40 | Male | 74000 | 4 | NULL |
| 5 | Pavi | 23 | Female | 76000 | 3 | NULL |
| 6 | Yuva | 29 | Male | 64900 | 1 | NULL |
| 7 | Divya | 25 | Female | 67000 | 3 | NULL |
| 8 | Gomathi | 23 | Female | 90200 | 1 | NULL |
| 9 | Manisha | 26 | Female | 58000 | 4 | NULL |
| 10 | Deepak | 48 | Male | 65000 | 3 | NULL |
| 11 | Vaishu | 55 | Female | 74000 | 3 | NULL |
| 12 | Anish | 36 | Male | 62700 | 1 | NULL |
| 13 | Vinitha | 49 | Female | 69000 | 3 | NULL |
| 14 | Deekshit | 55 | Male | 79000 | 5 | NULL |
| 15 | Dharani | 45 | Female | 85000 | 3 | NULL |

### ***2. Ensure `age` is always greater than 1 :***

ALTER TABLE Employees ADD CHECK (Age>18);

SELECT \* FROM Employees ;

**Output :**

| emp\_id | emp\_name | age | gender | salary | dep\_id | email |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | Akhil | 25 | Male | 55000 | 3 | NULL |
| 2 | Bavya | 20 | Female | 68000 | 5 | NULL |
| 3 | Anbu | 35 | Male | 59000 | 3 | NULL |
| 4 | Vinoth | 40 | Male | 74000 | 4 | NULL |
| 5 | Pavi | 23 | Female | 76000 | 3 | NULL |
| 6 | Yuva | 29 | Male | 64900 | 1 | NULL |
| 7 | Divya | 25 | Female | 67000 | 3 | NULL |
| 8 | Gomathi | 23 | Female | 90200 | 1 | NULL |
| 9 | Manisha | 26 | Female | 58000 | 4 | NULL |
| 10 | Deepak | 48 | Male | 65000 | 3 | NULL |
| 11 | Vaishu | 55 | Female | 74000 | 3 | NULL |
| 12 | Anish | 36 | Male | 62700 | 1 | NULL |
| 13 | Vinitha | 49 | Female | 69000 | 3 | NULL |
| 14 | Deekshit | 55 | Male | 79000 | 5 | NULL |
| 15 | Dharani | 45 | Female | 85000 | 3 | NULL |

### ***3. Add a default value 'Unknown' to `gender` if not specified :***

ALTER TABLE Employees MODIFY gender VARCHAR(20) DEFAULT 'Unknown' ;

### 

### 

## ***Case Study 2: Library Management System***

Scenario:

You are tasked with creating a database to manage a library's inventory, members, and borrowing records.

**MAIN SQL QUERY :**

CREATE DATABASE LibraryDB;

USE LibraryDB;

CREATE TABLE Books (id INT AUTO\_INCREMENT PRIMARY KEY,title VARCHAR(100) NOT NULL, author VARCHAR(50) NOT NULL,published\_year YEAR NOT NULL,genre VARCHAR(30));

INSERT INTO Books(title,author,published\_year,genre )VALUES

("Harper Lee" ,"To Kill a Mockingbird", 1960, "Thriller"),

("Dr. Seuss","Green Eggs and Ham", 1960,"Children's literature"),

("Joseph Heller","Catch-22", 1961," War story"),

("Ken Kesey","One Flew Over the Cuckoo's Nest",1962," Psychological Fiction"),

(" Madeline L' Engle","A Wrinkle in Time",1963,", Fantasy Fiction") ;

SELECT \* FROM Books;

CREATE TABLE Members (id INT AUTO\_INCREMENT PRIMARY KEY,name VARCHAR(50) NOT NULL,membership\_date DATE NOT NULL);

INSERT INTO Members (name,membership\_date) VALUES

('Anitha','2010-05-25'),

('Gomathi','2009-07-12'),

('Divya','2008-06-28'),

('Pavithra','2010-03-07'),

('Diya','2015-03-23');

SELECT \* FROM Members;

CREATE TABLE Borrowing (id INT AUTO\_INCREMENT PRIMARY KEY,book\_id INT,member\_id INT,borrow\_date DATE NOT NULL,return\_date DATE, FOREIGN KEY (book\_id) REFERENCES Books(id),FOREIGN KEY (member\_id) REFERENCES Members(id));

INSERT INTO Borrowing (book\_id,member\_id,borrow\_date,return\_date)VALUES

(1,3,'2023-02-10','2023-02-27'),

(3,2,'2023-05-17','2023-06-10'),

(2,5,'2023-09-22',NULL),

(5,1,'2024-03-23','2023-04-15'),

(4,4,'2024-08-06','2024-08-29');

SELECT \* FROM Borrowing;

## ***## Queries :***

### ***1.List all books :***

SELECT \* FROM Books;

**Output :**

| Id | Title | author | published\_year | genre |
| --- | --- | --- | --- | --- |
| 1 | Harper Lee | To Kill a Mockingbird | 1960 | Thriller |
| 2 | Dr. Seuss | Green Eggs and Ham | 1960 | Children's literature |
| 3 | Joseph Heller | Catch-22 | 1961 | War story |
| 4 | Ken Kesey | One Flew Over the Cuckoo's Nest | 1962 | Psychological Fiction |
| 5 | Madeline L' Engle | A Wrinkle in Time | 1963 | , Fantasy Fiction |

### 

### ***2. List all members :***

SELECT \* FROM Members ;

**Output :**

| Id | Name | Membership\_date |
| --- | --- | --- |
| 1 | Anitha | 2010-05-25 |
| 2 | Gomathi | 2009-07-12 |
| 3 | Divya | 2008-06-28 |
| 4 | Pavithra | 2010-03-07 |
| 5 | Diya | 2015-03-23 |

### ***3.Show borrowing records with member names and book titles :***

SELECT Borrowing.borrow\_date,Borrowing.return\_date,members.name, books.title FROM Borrowing JOIN members ON Borrowing.id=members.id JOIN books ON Borrowing.member\_id = books.id;

**Output :**

| borrow\_date | return\_date | name | title |
| --- | --- | --- | --- |
| 2024-03-23 | 2023-04-15 | Pavithra | Harper Lee |
| 2023-05-17 | 2023-06-10 | Gomathi | Dr. Seuss |
| 2024-08-06 | 2024-08-29 | Diya | Ken Kesey |
| 2023-09-22 | NULL | Divya | Madeline L' Engle |
| 2023-02-10 | 2023-02-27 | Anitha | Joseph Heller |

### ***4.Count the number of books borrowed by each member :***

SELECT Members.name, COUNT(Borrowing.id) FROM Members LEFT JOIN Borrowing ON Members.id = Borrowing.member\_id GROUP BY Members.id;

**Output :**

| Name | COUNT(Borrowing.id) |
| --- | --- |
| Anitha | 1 |
| Gomathi | 1 |
| Divya | 1 |
| Pavithra | 1 |
| Diya | 1 |

## 

## ***Case Study 3: Student Management System***

Scenario:

Create a database to manage students, their courses, and grades.

**MAIN SQL QUERY :**

CREATE DATABASE studentDB;

USE studentDB;

CREATE TABLE student (id INT AUTO\_INCREMENT PRIMARY KEY, name VARCHAR(50) NOT NULL, age INT NOT NULL, email VARCHAR(50)NOT NULL UNIQUE);

INSERT INTO student (name, age,email) VALUES

('Gomathi',22,'goms@123'),

('Pavithra',24,'pavi@123'),

('Divya',20,'dd@123'),

('Sharmi',23,'ammu@123'),

('Diya',25,'diya@123');

SELECT \* FROM student;

CREATE TABLE courses (id INT PRIMARY KEY AUTO\_INCREMENT, name VARCHAR(50) NOT NULL);

INSERT INTO courses (name) VALUES

('IT'),

('EEE'),

('ECE'),

('MECH');

SELECT \* FROM courses;

CREATE TABLE Enrollments (student\_id INT,course\_id INT,grade CHAR(1),PRIMARY KEY (student\_id, course\_id), FOREIGN KEY (student\_id) REFERENCES Students(id),FOREIGN KEY (course\_id) REFERENCES Courses(id));

INSERT INTO Enrollments (student\_id,course\_id,grade) VALUES

(1, 3, 'A'),

(2, 2, 'B'),

(3, 4, 'D'),

(4, 1, 'B'),

(5, 2, 'A');

SELECT \* FROM Enrollments;

## ***## Queries :***

### ***1.List all students:***

SELECT \* FROM student;

**Output :**

| id | name | age | email |
| --- | --- | --- | --- |
| 1 | Gomathi | 22 | goms@123 |
| 2 | Pavithra | 24 | pavi@123 |
| 3 | Divya | 20 | dd@123 |
| 4 | Sharmi | 23 | ammu@123 |
| 5 | Diya | 25 | diya@123 |

### ***2.List all courses:***

SELECT \* FROM courses;

**Output :**

| id | name |
| --- | --- |
| 1 | IT |
| 2 | EEE |
| 3 | ECE |
| 4 | MECH |

### 

### ***3.Show the names of students and the courses they are enrolled in :***

SELECT student.name AS stu\_name ,courses.name AS course\_name FROM student JOIN Enrollments ON student.id= Enrollments.student\_id JOIN Courses ON Enrollments.Course\_id = Courses.id;

**Output :**

| stu\_name | course\_name |
| --- | --- |
| Sharmi | IT |
| Pavithra | EEE |
| Diya | EEE |
| Gomathi | ECE |
| Divya | MECH |

### ***4.Count the number of students in each course :***

SELECT Courses.name as courses\_name,COUNT(Enrollments.student\_id) FROM Courses LEFT JOIN Enrollments ON Courses.id = Enrollments.course\_id GROUP BY Courses.id;

**Output :**

| courses\_name | COUNT(Enrollments.student\_id) |
| --- | --- |
| IT | 1 |
| EEE | 2 |
| ECE | 1 |
| MECH | 1 |

## ***Case Study 4 : Retail Store Inventory System***

Scenario:

Design a database to manage a retail store's inventory, suppliers, and sales.

**MAIN SQL QUERY :**

CREATE DATABASE RetailDB;

USE RetailDB ;

CREATE TABLE Products (id INT PRIMARY KEY AUTO\_INCREMENT,name VARCHAR(50) NOT NULL, price DECIMAL(10, 2) NOT NULL,CHECK (price >0),stock INT NOT NULL ,CHECK (stock>=0));

INSERT INTO products(name,price,stock) VALUES

('KIT KAT',10.00,5),

('Milk Bikis',20.00,10),

('ICE CREAM',25.50,15),

('BROWNIE CAKE',50.00,20),

('JUICE',40.00,10);

SELECT \* FROM products ;

CREATE TABLE Suppliers(id INT PRIMARY KEY AUTO\_INCREMENT,name VARCHAR(50) NOT NULL, contact VARCHAR(50));

INSERT INTO Suppliers (name,contact) VALUES

('sup\_1','123456'),

('sup\_2','234567'),

('sup\_3','345678');

SELECT \* FROM Suppliers ;

CREATE TABLE sales (id INT PRIMARY KEY AUTO\_INCREMENT,product\_id INT ,FOREIGN KEY (product\_id) REFERENCES products(id), date DATE NOT NULL, quantity INT NOT NULL,CHECK(quantity>0),total\_price DECIMAL(10,2) NOT NULL);

INSERT INTO sales (product\_id,date,quantity,total\_price) VALUES

(1,'2024-02-10',3,30),

(2,'2024-02-15',7,140),

(3,'2024-02-23',10,510),

(4,'2024-02-27',15,750),

(5,'2024-03-05',5,200);

SELECT \* FROM sales ;

## ***## Queries :***

### ***1. List all products :***

SELECT \* FROM products;

**Output :**

| id | name | price | stock |
| --- | --- | --- | --- |
| 1 | KIT KAT | 10.00 | 5 |
| 2 | Milk Bikis | 20.00 | 10 |
| 3 | ICE CREAM | 25.00 | 15 |
| 4 | BROWNIE CAKE | 50.00 | 20 |
| 5 | JUICE | 40.00 | 20 |

### 

### ***2. Show products with stock less than 10:***

SELECT \* FROM products WHERE stock<10;

**Output :**

| id | name | price | stock |
| --- | --- | --- | --- |
| 1 | KIT KAT | 10.00 | 5 |

### ***3.List sales with product names and total prices :***

SELECT products.name AS product\_name,total\_price,quantity,date FROM products JOIN sales

ON products.id = sales.id;

**Output :**

| product\_name | total\_price | quantity | date |
| --- | --- | --- | --- |
| KIT KAT | 30.00 | 3 | 2024-02-10 |
| Milk Bikis | 140.00 | 7 | 2024-02-15 |
| ICE CREAM | 510.00 | 10 | 2024-02-23 |
| BROWNIE CAKE | 750.00 | 15 | 2024-02-27 |
| JUICE | 200.00 | 5 | 2024-03-05 |

### ***4.Calculate the total sales for each product :***

SELECT products.name AS product\_name,SUM(sales.total\_price) AS total\_sales FROM products JOIN sales ON products.id = sales.id GROUP BY product\_name;

**Output :**

| product\_name | total\_sales |
| --- | --- |
| KIT KAT | 30.00 |
| Milk Bikis | 140.00 |
| ICE CREAM | 510.00 |
| BROWNIE CAKE | 750.00 |
| JUICE | 200.00 |

## **Case Study 5: Hospital Management System**

Scenario:

Create a database to manage patients, doctors, and appointments in a hospital.

**MAIN SQL QUERY :**

CREATE DATABASE HospitalDB;

USE HospitalDB;

CREATE TABLE Patients (id INT AUTO\_INCREMENT PRIMARY KEY, name VARCHAR(50) NOT NULL,age INT NOT NULL,gender VARCHAR(10),contact VARCHAR(50));

INSERT INTO patients (name,age,gender,contact) VALUES

('Akshaya',35,'Female','123456'),

('Akash',55,'Male','234567'),

('Tamil',40,'Female','345678'),

('Deena',78,'Male','456789'),

('Vinitha',35,'Female','567891');

SELECT \* FROM patients;

CREATE TABLE Doctors (id INT AUTO\_INCREMENT PRIMARY KEY, name VARCHAR(50) NOT NULL,specialization VARCHAR(50) NOT NULL);

INSERT INTO Doctors (name,specialization) VALUES

("Deekshit","Cardiologist"),

("Nani","Neurologist"),

("Darsha","General surgery");

SELECT \* FROM Doctors;

CREATE TABLE Appointments( id INT AUTO\_INCREMENT PRIMARY KEY, patient\_id INT,

FOREIGN KEY ( patient\_id) REFERENCES Patients(id), doctor\_id INT, FOREIGN KEY (doctor\_id) REFERENCES Doctors(id),

appointment\_date DATE NOT NULL, status ENUM ('Scheduled', 'Completed', 'Cancelled'));

INSERT INTO Appointments (patient\_id, doctor\_id, appointment\_date, status) VALUES

(1, 1, '2024-02-10', 'Scheduled'),

(2, 2, '2024-02-17', 'Completed'),

(3, 3, '2024-02-21', 'Cancelled'),

(4, 1, '2024-03-05', 'Scheduled'),

(5, 3, '2024-03-15', 'Completed');

SELECT \* FROM Appointments;

## ***## Queries :***

### ***1.List all patients :***

SELECT \* FROM patients;

**Output :**

| id | name | age | gender | contact |
| --- | --- | --- | --- | --- |
| 1 | Akshaya | 35 | Female | 123456 |
| 2 | Akash | 55 | Male | 234567 |
| 3 | Tamil | 40 | Female | 345678 |
| 4 | Deena | 78 | Male | 456789 |
| 5 | Vinitha | 35 | Female | 567891 |

### 

### ***2. List all doctors :***

SELECT \* FROM Doctors;

**Output :**

| id | name | specialization |
| --- | --- | --- |
| 1 | Deekshit | Cardiologist |
| 2 | Nani | Neurologist |
| 3 | Darsha | General surgery |

### ***3. Show appointments with patient and doctor names :***

SELECT Appointments.id AS appointment\_id,patients.name AS patient\_name,doctors.name AS doctor\_name,Appointments.appointment\_date, status FROM Appointments JOIN Patients ON Appointments.patient\_id = patients.id JOIN Doctors ON Appointments.doctor\_id = Doctors.id;

**Output :**

| appointment\_id | patient\_name | doctor\_name | appointment\_date | status |
| --- | --- | --- | --- | --- |
| 1 | Akshaya | Deekshit | 2024-02-10 | Scheduled |
| 4 | Deena | Deekshit | 2024-03-05 | Scheduled |
| 2 | Akash | Nani | 2024-02-17 | Completed |
| 3 | Tamil | Darsha | 2024-02-21 | Cancelled |
| 5 | Vinitha | Darsha | 2024-03-15 | Completed |

### ***4. Count appointments per doctor :***

SELECT Doctors.name AS doctor\_name,COUNT( Appointments.id) AS appointment\_count FROM Doctors LEFT JOIN Appointments ON Doctors.id = Appointments.doctor\_id GROUP BY Doctors.id ;

**Output :**

| doctor\_name | appointment\_count |
| --- | --- |
| Deekshit | 2 |
| Nani | 1 |
| Darsha | 2 |