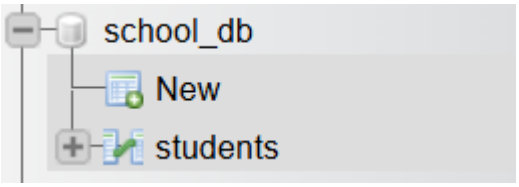
Module 4 – Introduction to DBMS

1. Introduction to SQL

LAB EXERCISES:

1) Create a new database named school_db and a table called students with the following columns: student_id, student_name, age, class, and address.



student_id	student_name	age	class	address

2) Insert five records into the students table and retrieve all records using the SELECT statement.

	student_id	student_name	age	class	address
Edit Copy Delete	1	Jil	22	8	Vastral
Edit Copy Delete	2	shail	24	10	Maninager
Edit Copy Delete	3	Dhruvin	21	9	Ghodaser
Edit Copy Delete	4	Om	19	7	Gota
Edit Copy Delete	5	Mangalam	20	6	Vadodra

Showing rows 0 - 4 (5 total, Query took 0.0002 seconds.)

```
SELECT student_id, student_name, age, class, address FROM students WHERE 1;
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 Filter rows: Search this table Sort

Extra options

	student_id	student_name	age	class	address
Edit Copy Delete	1	Jil	22	8	Vastral
Edit Copy Delete	2	shail	24	10	Maninager
Edit Copy Delete	3	Dhruvin	21	9	Ghodaser
Edit Copy Delete	4	Om	19	7	Gota
Edit Copy Delete	5	Mangalam	20	6	Vadodra

2. SQL Syntax

1) Write SQL queries to retrieve specific columns (student_name and age) from the students table.

```
SELECT student_name, age FROM students WHERE 1;
```

	student_name	age
Edit Copy Delete	Jil	22
Edit Copy Delete	shail	24
Edit Copy Delete	Dhruvin	21
Edit Copy Delete	Om	19
Edit Copy Delete	Mangalam	20

2) Write SQL queries to retrieve all students whose age is greater than 10.

```
SELECT age FROM students WHERE age > 10;
```

	age
Edit Copy Delete	19
Edit Copy Delete	20
Edit Copy Delete	21
Edit Copy Delete	22
Edit Copy Delete	24

3. SQL Constraints

1) Create a table teachers with the following columns: teacher_id (Primary Key), teacher_name (NOT NULL), subject (NOT NULL), and email (UNIQUE).

```
CREATE TABLE Teachers ( teacher_id INT PRIMARY KEY, teacher_name VARCHAR(50) NOT NULL, subject VARCHAR(60) NOT NULL, email VARCHAR(70) UNIQUE );
```

teachers

teacher_id	teacher_name	subject	email

2) Implement a FOREIGN KEY constraint to relate the teacher_id fromthe teachers table with the students table.

```
ALTER TABLE teachers ADD CONSTRAINT fk_students_teachers FOREIGN KEY (teacher_id) REFERENCES students(student_id);
```

4. Main SQL Commands and Sub-commands (DDL)

1) Create a table courses with columns: course_id, course_name, and course_credits. Set the course_id as the primary key.

```
CREATE TABLE courses( course_id int PRIMARY KEY , course_name varchar(60), course_credits varchar(80) );
```

course_id	course_name	course_credits

2) Use the CREATE command to create a database university_db

```
CREATE DATABASE university_db;
```

5. ALTER Command

1) Modify the courses table by adding a column course_duration using the ALTER command.
ALTER TABLE courses ADD COLUMN course_duration int;

course_id	course_name	course_credits	course_duration

2) Drop the course_credits column from the courses table.

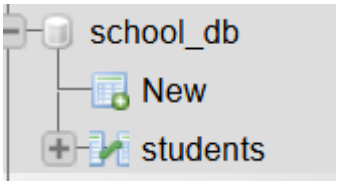
```
ALTER TABLE courses DROP COLUMN course_credits;
```

course_id	course_name	course_duration

6. DROP Command

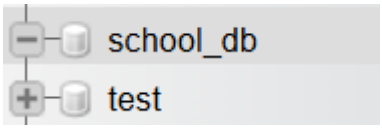
1) Drop the teachers table from the school_db database.

```
DROP TABLE teachers;
```



2) Drop the students table from the school_db database and verify that the table has been removed.

```
DROP TABLE students;
```



7. Data Manipulation Language (DML)

1) Insert three records into the courses table using the INSERT command.
INSERT INTO courses (course_id, course_name, course_duration) VALUES (1,"C Basic","3 month"),(2,"C++ Basic","1 month"),(3,"DBMS ","2 month");

	course_id	course_name	course_duration
Edit Copy Delete	1	C Basic	3
Edit Copy Delete	2	C++ Basic	1
Edit Copy Delete	3	DBMS	2

2) Update the course duration of a specific course using the UPDATE command.

```
UPDATE courses SET course_duration = '4' WHERE course_id = 2;
```

	course_id	course_name	course_duration
Edit Copy Delete	1	C Basic	3
Edit Copy Delete	2	C++ Basic	4
Edit Copy Delete	3	DBMS	2

3) Delete a course with a specific course_id from the courses table using the DELETE command.

```
DELETE FROM courses WHERE course_id = 3;
```

	course_id	course_name	course_duration
Edit Copy Delete	1	C Basic	3
Edit Copy Delete	2	C++ Basic	4

8. Data Query Language (DQL)

1) Retrieve all courses from the courses table using the SELECT statement.

```
SELECT * FROM courses;
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 Filter rows: Search this table

Extra options

	course_id	course_name	course_duration
Edit Copy Delete	1	C Basic	3
Edit Copy Delete	2	C++ Basic	4

2) Sort the courses based on course_duration in descending order using ORDERBY.

```
SELECT * FROM courses ORDER BY course_duration DESC;
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 Filter rows: Search this table

Extra options

	course_id	course_name	course_duration ↓ 1
Edit Copy Delete	2	C++ Basic	4
Edit Copy Delete	1	C Basic	3

3) Limit the results of the SELECT query to show only the top two courses using LIMIT.

```
SELECT * FROM courses LIMIT 2;
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Extra options

	course_id	course_name	course_duration
Edit Copy Delete	1	C Basic	3
Edit Copy Delete	2	C++ Basic	4

9. Data Control Language (DCL)

1) Create two new users user1 and user2 and grant user1 permission to SELECT from the courses table.

```
CREATE USER user1 IDENTIFIED BY 'your_password';
```

```
GRANT SELECT ON courses TO user1;
```

```
CREATE USER user2 IDENTIFIED BY 'your_password';
```

2) Revoke the INSERT permission from user1 and give it to user2.

```
REVOKE INSERT ON courses FROM user1;
```

```
GRANT INSERT ON courses TO user2;
```

10. Transaction Control Language (TCL)

1) Insert a few rows into the courses table and use COMMIT to save the changes.

```
SELECT * FROM courses LIMIT 2;
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Extra options

	course_id	course_name	course_duration
Edit Copy Delete	1	C Basic	3
Edit Copy Delete	2	C++ Basic	4

```
INSERT INTO courses(course_id, course_name, course_duration) VALUES (3,"Introduction to SQL",'2'),(4,"Advanced SQL Techniques",'1');
```

```
COMMIT;
```

course_id	course_name	course_duration
1	C Basic	3
2	C++ Basic	4
3	Introduction to SQL	2
4	Advanced SQL Techniques	1

2) Insert additional rows, then use ROLLBACK to undo the last insert operation.
INSERT INTO courses(course_id, course_name, course_duration) VALUES
(5,"Web Development Fundamentals",5),
(6,"Mobile App Development with Java",6);
ROLLBACK;

```
INSERT INTO courses(course_id, course_name, course_duration) VALUES (5,"Web Development Fundamentals",5), (6,"Mobile App Development with Java",6);
```

```
ROLLBACK;
```

course_id	course_name	course_duration
1	C Basic	3
2	C++ Basic	4
3	Introduction to SQL	2
4	Advanced SQL Techniques	1
5	Web Development Fundamentals	5
6	Mobile App Development with Java	6

3) Create a SAVEPOINT before updating the courses table, and use it to roll back specific changes.

```
START TRANSACTION;
```

```
INSERT INTO courses(course_id, course_name, course_duration) VALUES (7,"Introduction to Machine Learning",5);
```

```
SAVEPOINT update_start;
```

```
UPDATE courses SET course_duration = 9 WHERE course_name = 'Introduction to SQL';
```

```
INSERT INTO courses(course_id, course_name, course_duration) VALUES (8,"Data Visualization with Tableau",10);
```

```
ROLLBACK TO update_start;
```

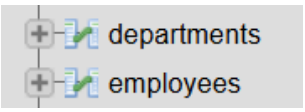
```
COMMIT;
```

course_id	course_name	course_duration
1	C Basic	3
2	C++ Basic	4
3	Introduction to SQL	2
4	Advanced SQL Techniques	1
5	Web Development Fundamentals	5
6	Mobile App Development with Java	6
7	Introduction to Machine Learning	5

11. SQL Joins

1) Create two tables: departments and employees. Perform an INNER JOIN to display employees along with their respective departments.

```
CREATE TABLE departments( department_name varchar (100) PRIMARY KEY );
```



←T→		eid	ename	efid
Edit Copy Delete		1	jil	1
Edit Copy Delete		2	dhruvin	2
Edit Copy Delete		3	zainil	3
Edit Copy Delete		4	shubham	4

←T→		d_id	d_name
Edit Copy Delete		1	Head Department
Edit Copy Delete		2	senior Department
Edit Copy Delete		3	junior Department
Edit Copy Delete		4	base Department

```
SELECT departments.d_name,employees.ename FROM departments INNER JOIN employees ON departments.d_id = employees.efid;
```

d_name	ename
Head Department	jil
senior Department	dhruvin
junior Department	zainil
base Department	shubham

2) Use a LEFT JOIN to show all departments, even those without employees.

```
SELECT departments.d_name FROM departments LEFT JOIN employees ON departments.d_id = employees.efid;
```

d_name
Head Department
senior Department
junior Department
base Department

12. SQL Group By

1) Group employees by department and count the number of employees in each department using GROUP BY.

```
SELECT departments.d_name,COUNT(employees.eid) AS employeesCoun FROM departments LEFT JOIN employees ON departments.d_id = Employees.efid GROUP BY departments.d_name;
```

d_name	employeesCoun
base Department	1
Head Department	1
junior Department	1
senior Department	1

2) Use the AVG aggregate function to find the average salary of employees in each department.

```
SELECT AVG(salary) FROM employees;
```

Profiling [Edit inline] [Edit] [Explain SQL

Show all | Number of rows: 25

Extra options

AVG(salary)
19000.0000

13. SQL Stored Procedure
    1) Write a stored procedure to retrieve all employees from the employees table based on department.

```
DELIMITER $$
CREATE PROCEDURE employee()
BEGIN
SELECT * FROM employees;
END
```

```
CALL employee();
```

[Edit inline] [Edit] [Create PHP co

Show all | Number of rows

Extra options

eid	ename	efid	salary
1	jil	1	22000
2	dhruvin	2	20000
3	zainil	3	16000
4	shubham	4	18000

    2) Write a stored procedure that accepts course_id as input and returns the course details.

←T→	course_id	course_name	course_duration
Edit Copy Delete	1	C Basic	3
Edit Copy Delete	2	C++ Basic	4

```
CREATE PROCEDURE Course() BEGIN SELECT course_name FROM courses WHERE course_id = '1'; END;;
```

```
CALL Course;
```

[Edit inline] [Edit] [C

Show all | N

Extra options

course_name
C Basic

14. SQL View

1) Create a view to show all employees along with their department names.

```
CREATE VIEW EmployeeWithDepartment AS SELECT employees.eid,employees.ename,departments.d_name FROM employees JOIN departments ON employees.eid= departments.d_id;
```

```
SELECT * FROM EmployeeWithDepartment;
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refre

Show all | Number of rows: 25 Filter rows: Search

Extra options

		eid	ename	d_name
Edit Copy Delete		1	jil	Head Department
Edit Copy Delete		2	dhruvin	senior Department
Edit Copy Delete		3	zainil	junior Department
Edit Copy Delete		4	shubham	base Department

2) Modify the view to exclude employees whose salaries are below $50,000.

```
CREATE VIEW EmployeeDepartment AS SELECT employees.eid,employees.ename,departments.d_id,employees.salary FROM employees JOIN departments ON employees.eid = departments.d_id
WHERE employees.salary < 20000;
```

```
SELECT * FROM EmployeeDepartment;
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refre

Show all | Number of rows: 25 Filter rows: Search

Extra options

		eid	ename	d_id	salary
Edit Copy Delete		3	zainil	3	16000
Edit Copy Delete		4	shubham	4	18000

15. SQL Triggers

1) Create a trigger to automatically log changes to the employees table when a new employee is added.

```
CREATE TABLE trigger_table(tid int,
                tname varchar(80),
                tsalary bigint
                );
```

CREATE TABLE employee_log (log_id INT PRIMARY KEY AUTO_INCREMENT, employee_id INT, action VARCHAR(50), timestamp DATETIME DEFAULT CURRENT_TIMESTAMP);

log_id	employee_id	action	timestamp

```
DELIMITER //
CREATE TRIGGER log_new_employee
AFTER INSERT ON employees
FOR EACH ROW
BEGIN
  INSERT INTO employee_log (eid, action)
  VALUES (NEW.eid, 'New employee added');
END;
//
DELIMITER ;
```

```
CREATE TRIGGER log_new_employee AFTER INSERT ON employees FOR EACH ROW BEGIN INSERT INTO employee_log (eid, action) VALUES (NEW.eid, 'New employee added'); END;;
```

2) Create a trigger to update the last_modified timestamp whenever an employee record is updated.

```
ALTER TABLE employees ADD COLUMN last_modified DATETIME;
```

```
CREATE TRIGGER update_employee_last_modified BEFORE UPDATE ON employees FOR EACH ROW BEGIN SET NEW.last_modified = CURRENT_TIMESTAMP; END;;
```

eid	ename	efid	salary	last_modified
1	jil	1	22000	*NULL*
2	dhruvin	2	20000	*NULL*
3	zainil	3	16000	*NULL*
4	shubham	4	18000	*NULL*

```
UPDATE employees SET ename = 'Jane Doe' WHERE eid = 2;
```

eid	ename	efid	salary	last_modified
2	Jane Doe	2	20000	2025-04-10 17:51:20

16. Introduction to PL/SQL

1) Write a PL/SQL block to print the total number of employees from the employee stable.

```
DECLARE
 total_employees NUMBER;
BEGIN
 SELECT COUNT(*) INTO total_employees FROM employees;
 DBMS_OUTPUT.PUT_LINE('Total number of employees: ' || total_employees);
```

```
END;
/
```

2) Create a PL/SQL block that calculates the total sales from an orders table.

17. PL/SQL Control Structures

1) Write a PL/SQL block using an IF-THEN condition to check the department of an employee.

```
DECLARE
 eid NUMBER := 100; -- Replace with the employee ID you want to check
 departments VARCHAR2(20);
BEGIN
 SELECT departments INTO ename FROM employees WHERE eid = eid;

 IF employee_dept = 'Sales' THEN
  DBMS_OUTPUT.PUT_LINE('Employee is in the Sales department.');
 ELSIF employee_dept = 'Marketing' THEN
  DBMS_OUTPUT.PUT_LINE('Employee is in the Marketing department.');
 ELSE
  DBMS_OUTPUT.PUT_LINE('Employee is in a different department.');
 END IF;
END;
/
```

2) Use a FOR LOOP to iterate through employee records and display their names.

```
DECLARE
 employee_record employees%ROWTYPE;
BEGIN
 FOR employee_record IN (SELECT * FROM employees) LOOP
  DBMS_OUTPUT.PUT_LINE('Employee Name: ' ||
employee_record.first_name || ' ' || employee_record.last_name);
 END LOOP;
END;
/
```

18. SQL Cursors

1) Write a PL/SQL block using an explicit cursor to retrieve and display employee details.

```
DECLARE
 -- Declare the cursor
 CURSOR employee_cursor IS
  SELECT employee_id, first_name, last_name, department_id
  FROM employees;

 -- Declare variables to store employee details
```

```
 employee_id_var NUMBER;
 first_name_var VARCHAR2(20);
 last_name_var VARCHAR2(25);
 department_id_var NUMBER;
BEGIN
 -- Open the cursor
 OPEN employee_cursor;

 -- Fetch data from the cursor
 LOOP
  FETCH employee_cursor INTO employee_id_var, first_name_var, last_name_var, department_id_var;
  EXIT WHEN employee_cursor%NOTFOUND;

  -- Display employee details
  DBMS_OUTPUT.PUT_LINE('Employee ID: ' || employee_id_var);
  DBMS_OUTPUT.PUT_LINE('First Name: ' || first_name_var);
  DBMS_OUTPUT.PUT_LINE('Last Name: ' || last_name_var);
  DBMS_OUTPUT.PUT_LINE('Department ID: ' || department_id_var);
  DBMS_OUTPUT.PUT_LINE('-----------------------');
 END LOOP;

 -- Close the cursor
 CLOSE employee_cursor;
END;
/
```

2) Create a cursor to retrieve all courses and display them one by one.

```
DECLARE
 -- Declare the cursor
 CURSOR course_cursor IS
  SELECT course_name, course_description, enrollment
  FROM courses;

 -- Declare variables to store course details
 course_name_var VARCHAR2(100);
 course_description_var VARCHAR2(200);
 enrollment_var NUMBER;
BEGIN
 -- Open the cursor
 OPEN course_cursor;

 -- Fetch data from the cursor
 LOOP
  FETCH course_cursor INTO course_name_var, course_description_var, enrollment_var;
  EXIT WHEN course_cursor%NOTFOUND;

  -- Display course details
  DBMS_OUTPUT.PUT_LINE('Course Name: ' || course_name_var);
  DBMS_OUTPUT.PUT_LINE('Description: ' || course_description_var);
  DBMS_OUTPUT.PUT_LINE('Enrollment: ' || enrollment_var);
  DBMS_OUTPUT.PUT_LINE('-----------------------');
 END LOOP;

 -- Close the cursor
 CLOSE course_cursor;
```

```
END;
/
```

19. Rollback and Commit Savepoint

1) Perform a transaction where you create a savepoint, insert records, then rollback to the savepoint.

```
BEGIN TRANSACTION;

INSERT INTO employees (eid, ename, efid) VALUES (6, "Alice", '6');
INSERT INTO employees (eid, ename, efid) VALUES (7, "Bob", '7');

SAVEPOINT my_savepoint;

INSERT INTO employees (eid, ename, efid) VALUES (8, "charlie", '8');
INSERT INTO employees (eid, ename, efid) VALUES (9, "david", '9');

ROLLBACK TO SAVEPOINT my_savepoint;

COMMIT;
```

2) Commit part of a transaction after using a savepoint and then rollback the remaining changes.

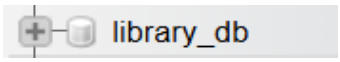
```
BEGIN TRANSACTION;

INSERT INTO employees (eid, ename, efid) VALUES (9, "Alice", '9');

INSERT INTO employees (eid, ename, efid) VALUES (10, "Bob", '10');

SAVEPOINT my_savepoint;

INSERT INTO employees (eid, ename, efid) VALUES (11, "Charlie", '11');

INSERT INTO employees (eid, ename, efid) VALUES (12, "David", '12');

RELEASE SAVEPOINT my_savepoint;

COMMIT;


BEGIN TRANSACTION;

INSERT INTO employees (eid, ename, efid) VALUES (19, "Eve", '13');

ROLLBACK;

COMMIT;
```

EXTRA LAB PRACTISE FOR DATABASE CONCEPTS

1. Introduction to SQL

3 Create a database called library_db and a table books with columns: book_id, title, author, publisher, year_of_publication, and price. Insert five records into the table.

```
CREATE DATABASE library_db;
```




```
CREATE TABLE books( book_id int PRIMARY KEY, title varchar (100), author varchar (100), publisher varchar (100), year_of_publication int , price int );
```

	book_id	title	author	publisher	year_of_publication	price
Edit Copy Delete	1	Making India Awesome	Chetan Bhagat	studio	1990	500
Edit Copy Delete	2	One Indian Girl	Chetan N Bhagat	first studio	1991	600
Edit Copy Delete	3	Revolution 2020	Balwant Gargi	second studio	1995	700
Edit Copy Delete	4	A Million Mutinies Now	V.S. Naipaul	nine studio	1996	900
Edit Copy Delete	5	A Brush with Life	Satish Gujral	only studio	2000	1000

4) Create a table members in library_db with columns: member_id, member_name, date_of_membership, and email. Insert five records into this table.

```
CREATE TABLE members( member_id int PRIMARY KEY, member_name varchar (100), date_of_membership int, email varchar (90) UNIQUE KEY );
```

	member_id	member_name	date_of_membership	email
Edit Copy Delete	1	Jil	2000	jp24@gmail.com
Edit Copy Delete	2	Dhruvin	2021	dp15@gmail.com
Edit Copy Delete	3	Zainil	2024	zp2@gmail.com
Edit Copy Delete	4	Shubham	2023	sp11@gmail.com
Edit Copy Delete	5	Rushi	2020	rp33@gmail.com

2. SQL Syntax

3 Retrieve all members who joined the library before 2022. Use appropriate SQL syntax with WHERE and ORDER BY.

WHERE

```
SELECT * FROM members WHERE date_of_membership < 2022;
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None

Extra options

	member_id	member_name	date_of_membership	email
Edit Copy Delete	1	Jil	2000	jp24@gmail.com
Edit Copy Delete	2	Dhruvin	2021	dp15@gmail.com
Edit Copy Delete	5	Rushi	2020	rp33@gmail.com

ORDER BY

```
SELECT * FROM members WHERE date_of_membership < 2022 GROUP BY date_of_membership ORDER BY date_of_membership ASC;
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None

Extra options

	member_id	member_name	date_of_membership ▲ 1	email
Edit Copy Delete	1	Jil	2000	jp24@gmail.com
Edit Copy Delete	5	Rushi	2020	rp33@gmail.com
Edit Copy Delete	2	Dhruvin	2021	dp15@gmail.com

4 Write SQL queries to display the titles of books published by a specific author. Sort the results by year_of_publication in descending order.

```
SELECT title, author FROM books WHERE author = "Chetan Bhagat";
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 Filter rows: Search this table

Extra options

	title	author
Edit Copy Delete	Making India Awesome	Chetan Bhagat

year_of_publication in descending order

```
SELECT * FROM books ORDER BY year_of_publication DESC;
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None

Extra options

	book_id	title	author	publisher	year_of_publication ▼ 1	price
Edit Copy Delete	5	A Brush with Life	Satish Gujral	only studio	2000	1000
Edit Copy Delete	4	A Million Mutinies Now	V.S. Naipaul	nine studio	1996	900
Edit Copy Delete	3	Revolution 2020	Balwant Gargi	second studio	1995	700
Edit Copy Delete	2	One Indian Girl	Chetan N Bhagat	first studio	1991	600
Edit Copy Delete	1	Making India Awesome	Chetan Bhagat	studio	1990	500

3. SQL Constraints

3 Add a CHECK constraint to ensure that the price of books in the books table is greater than 0.

```
ALTER TABLE books ADD CONSTRAINT CK_BookPrice CHECK (price > 0);
```

4 Modify the members table to add a UNIQUE constraint on the email column, ensuring that each member has a unique email address.

```
ALTER TABLE members ADD CONSTRAINT uq_memberemail UNIQUE (email);
```

4. Main SQL Commands and Sub-commands (DDL)

3) Create a table authors with the following columns: author_id, first_name, last_name, and country. Set author_id as the primary key.

```
CREATE TABLE authors( author_id int PRIMARY KEY AUTO_INCREMENT, first_name varchar(80), last_name varchar(80), country varchar(80) );
```

author_id	first_name	last_name	country

4) Create a table publishers with columns: publisher_id, publisher_name, contact_number, and address. Set publisher_id as the primary key and contact_number as unique.

```
CREATE TABLE publishers( publisher_id int PRIMARY KEY AUTO_INCREMENT, publisher_name varchar(80), contact_number bigint UNIQUE KEY, address varchar(100) );
```

publisher_id	publisher_name	contact_number	address

5. ALTER Command

3) Add a new column genre to the books table. Update the genre for all existing records.

```
ALTER TABLE books ADD COLUMN genre varchar (80);
```

```
UPDATE books SET genre = 'Fiction' WHERE title = 'Revolution 2020';
```

book_id	title	author	publisher	year_of_publication	price	genre
1	Making India Awesome	Chetan Bhagat	studio	1990	500	*NULL*
2	One Indian Girl	Chetan N Bhagat	first studio	1991	600	*NULL*
3	Revolution 2020	Balwant Gargi	second studio	1995	700	Fiction
4	A Million Mutinies Now	V.S. Naipaul	nine studio	1996	900	*NULL*
5	A Brush with Life	Satish Gujral	only studio	2000	1000	Fiction

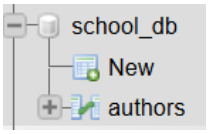
4) Modify the members table to increase the length of the email column to 100 characters.

```
ALTER TABLE members MODIFY email VARCHAR(100);
```

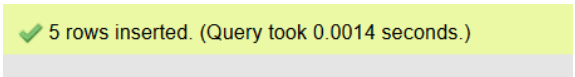
6. DROP Command

3) Drop the publishers table from the database after verifying its structure.

```
DROP TABLE publishers;
```



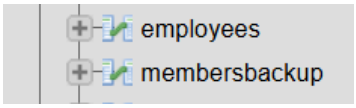
4) Create a backup of the members table and then drop the original members table.




```
INSERT INTO membersbackup SELECT * FROM members;
```

	member_id	member_name	date_of_membership	email
Edit Copy Delete	2	Dhruvin	2021	dp15@gmail.com
Edit Copy Delete	3	Zainil	2024	zp2@gmail.com
Edit Copy Delete	4	Shubham	2023	sp11@gmail.com
Edit Copy Delete	5	Rushi	2020	rp33@gmail.com
Edit Copy Delete	6	jack	2015	jacl01@gmail.com

```
DROP TABLE members;
```



7. Data Manipulation Language (DML)

5 Insert three new authors into the authors table, then update the last name of one of the authors.

	author_id	first_name	last_name
Edit Copy Delete	1	jack	boss
Edit Copy Delete	2	sam	jack
Edit Copy Delete	3	sarah	peterson

```
UPDATE authors SET last_name = 'Brown' WHERE first_name = 'sam' AND last_name = 'jack';
```

	author_id	first_name	last_name	country
Edit Copy Delete	1	jack	boss	*NULL*
Edit Copy Delete	2	sam	Brown	*NULL*
Edit Copy Delete	3	sarah	peterson	*NULL*

6 Delete a book from the books table where the price is higher than $100.

book_id	title	author	publisher	year_of_publication	price	genre
1	Making India Awesome	Chetan Bhagat	studio	1990	500	NULL
2	One Indian Girl	Chetan N Bhagat	first studio	1991	600	NULL
3	Revolution 2020	Balwant Gargi	second studio	1995	700	Fiction
4	A Million Mutinies Now	V.S. Naipaul	nine studio	1996	900	NULL
5	A Brush with Life	Satish Gujral	only studio	2000	1000	Fiction

Delete higher then 900

```
DELETE FROM books WHERE price > 900;
```

book_id	title	author	publisher	year_of_publication	price	genre
1	Making India Awesome	Chetan Bhagat	studio	1990	500	NULL
2	One Indian Girl	Chetan N Bhagat	first studio	1991	600	NULL
3	Revolution 2020	Balwant Gargi	second studio	1995	700	Fiction
4	A Million Mutinies Now	V.S. Naipaul	nine studio	1996	900	NULL

8. UPDATE Command

3 Update the year_of_publication of a book with a specific book_id. UPDATE books SET year_of_publication = new_year WHERE book_id = your_book_id;

book_id	title	author	publisher	year_of_publication	price	genre
1	Making India Awesome	Chetan Bhagat	studio	1990	500	NULL
2	One Indian Girl	Chetan N Bhagat	first studio	1991	600	NULL
3	Revolution 2020	Balwant Gargi	second studio	1995	700	Fiction
4	A Million Mutinies Now	V.S. Naipaul	nine studio	1996	900	NULL

```
UPDATE books SET year_of_publication = 2021 WHERE book_id = '1';
```

book_id	title	author	publisher	year_of_publication	price	genre
1	Making India Awesome	Chetan Bhagat	studio	2021	500	NULL
2	One Indian Girl	Chetan N Bhagat	first studio	1991	600	NULL
3	Revolution 2020	Balwant Gargi	second studio	1995	700	Fiction
4	A Million Mutinies Now	V.S. Naipaul	nine studio	1996	900	NULL

4 Increase the price of all books published before 2015 by 10%.

book_id	title	author	publisher	year_of_publication	price	genre
1	Making India Awesome	Chetan Bhagat	studio	2021	500	NULL
2	One Indian Girl	Chetan N Bhagat	first studio	1991	600	NULL
3	Revolution 2020	Balwant Gargi	second studio	1995	700	Fiction
4	A Million Mutinies Now	V.S. Naipaul	nine studio	1996	900	NULL

```
UPDATE books SET price = price * 15 WHERE year_of_publication < 2015;
```

book_id	title	author	publisher	year_of_publication	price	genre
1	Making India Awesome	Chetan Bhagat	studio	2021	500	NULL
2	One Indian Girl	Chetan N Bhagat	first studio	1991	9900	NULL
3	Revolution 2020	Balwant Gargi	second studio	1995	11550	Fiction
4	A Million Mutinies Now	V.S. Naipaul	nine studio	1996	14850	NULL

9. DELETE Command

3 Remove all members who joined before 2020 from the members table.

member_id	member_name	date_of_membership	email
1	Jil	2000	jp24@gmail.com
2	Dhruvin	2021	dp15@gmail.com
3	Zainil	2024	zp2@gmail.com
4	Shubham	2023	sp11@gmail.com
5	Rushi	2020	rp33@gmail.com

```
DELETE FROM members WHERE date_of_membership < '2020';
```

member_id	member_name	date_of_membership	email
2	Dhruvin	2021	dp15@gmail.com
3	Zainil	2024	zp2@gmail.com
4	Shubham	2023	sp11@gmail.com
5	Rushi	2020	rp33@gmail.com

4) Delete all books that have a NULL value in the author column.

book_id	title	author	publisher	year_of_publication	price	genre
1	Making India Awesome	Chetan Bhagat	studio	2021	500	NULL
2	One Indian Girl	Chetan N Bhagat	first studio	1991	9900	NULL
3	Revolution 2020	Balwant Gargi	second studio	1995	11550	Fiction
4	A Million Mutinies Now	V.S. Naipaul	nine studio	1996	14850	NULL

```
DELETE FROM books WHERE author IS NULL;
```

book_id	title	author	publisher	year_of_publication	price	genre
1	Making India Awesome	Chetan Bhagat	studio	2021	500	NULL
3	Revolution 2020	Balwant Gargi	second studio	1995	11550	Fiction

10. Data Query Language (DQL)

3) Write a query to retrieve all books with price between $50 and $100.

book_id	title	author	publisher	year_of_publication	price	genre
1	Making India Awesome	Chetan Bhagat	studio	2021	500	NULL
2	one indian girl	chetan n bhagat	first studio	1991	800	Love story
3	Revolution 2020	Balwant Gargi	second studio	1995	11550	Fiction

Price between 100- 1000:

```
SELECT * FROM books WHERE price BETWEEN 100 AND 1000;
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None

Extra options

	book_id	title	author	publisher	year_of_publication	price	genre
Edit Copy Delete	1	Making India Awesome	Chetan Bhagat	studio	2021	500	*NULL*
Edit Copy Delete	2	one indian girl	chetan n bhagat	first studio	1991	800	Love story

4) Retrieve the list of books sorted by author in ascending order and limit the results to the top 3 entries.

```
SELECT books.book_id, books.title, auther.auther_name FROM books JOIN auther ON books.book_id = auther.author_id ORDER BY auther.auther_name ASC LIMIT 3;
```

book_id	title	auther_name ▲ 1
1	Making India Awesome	jack
2	one indian girl	paul
3	Revolution 2020	sarah

11. Data Control Language (DCL)

3 Grant SELECT permission to a user named librarian on the books table.

```
GRANT SELECT ON books TO librarian;
```

4) Grant INSERT and UPDATE permissions to the user admin on the members table.

- Members table is drop in another question. So I used membersbackup table.

```
GRANT INSERT, UPDATE ON membersbackup TO admin;
```

12. REVOKE Command

3) Revoke the INSERT privilege from the user librarian on the books table.

```
REVOKE INSERT ON books FROM librarian;
```

4) Revoke all permissions from user admin on the members table.
Members table is drop in another question. So I used membersbackup table.

```
REVOKE ALL PRIVILEGES ON membersbackup FROM admin;
```

13. Transaction Control Language (TCL)

3) Use COMMIT after inserting multiple records into the books table, then make another insertion and perform a ROLLBACK.

```
START TRANSACTION;
```

```
INSERT INTO books (book_id, title, author, year_of_publication) VALUES (5, 'Book One', 'Author A', 2021), (6, 'Book Two', 'Author B', 2022), (7, 'Book Three', 'Author C', 2023);
```

```
COMMIT;
```

```
INSERT INTO books (book_id, title, author, year_of_publication) VALUES (8, 'Book Four', 'Author D', 2024);
```

```
ROLLBACK;
```

book_id	title	author	publisher	year_of_publication	price	genre
1	Making India Awesome	Chetan Bhagat	studio	2021	500	NULL
2	one indian girl	chetan n bhagat	first studio	1991	800	Love story
3	Revolution 2020	Balwant Gargi	second studio	1995	11550	Fiction
5	Book One	Author A	NULL	2021	NULL	NULL
6	Book Two	Author B	NULL	2022	NULL	NULL
7	Book Three	Author C	NULL	2023	NULL	NULL
8	Book Four	Author D	NULL	2024	NULL	NULL

4) Set a SAVEPOINT before making updates to the members table, perform some updates, and then roll back to the SAVEPOINT.

```
START TRANSACTION;
```

```
SAVEPOINT before_updates;
```

```
UPDATE membersbackup SET status = 'active' WHERE member_id = 2;
```

```
UPDATE membersbackup SET status = 'inactive' WHERE member_id = 3;
```

```
ROLLBACK TO before_updates;
```

```
COMMIT;
```

member_id	member_name	date_of_membership	email	status
2	Dhruvin	2021	dp15@gmail.com	NULL
3	Zainil	2024	zp2@gmail.com	NULL
4	Shubham	2023	sp11@gmail.com	NULL
5	Rushi	2020	rp33@gmail.com	NULL
6	jack	2015	jacl01@gmail.com	NULL

14. SQL Joins

3) Perform an INNER JOIN between books and authors tables to display the title of books and their respective authors' names.

```
SELECT books.title,auther.auther_name FROM auther INNER JOIN books ON books.book_id = auther.auther_fid;
```

title	auther_name
Making India Awesome	jack
one indian girl	paul
Revolution 2020	sarah

4) Use a FULL OUTER JOIN to retrieve all records from the books and authors tables, including those with no matching entries in the other table.

SELECT * FROM books FULL OUTER JOIN auther ON books.book_id = auther.auther_id;

15. SQL Group By

3) Group books by genre and display the total number of books in each genre.

```
SELECT genre, COUNT(*) total_books FROM books GROUP BY genre ORDER BY genre;
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 Filter rows: Search this table

Extra options

genre ▲ 1	total_books
NULL	1
Fiction	1
Love story	1

4) Group members by the year they joined and find the number of members who joined each year.

```
SELECT date_of_membership AS JoiningYear, COUNT(*) AS NumberOfMembers FROM members GROUP BY date_of_membership ORDER BY JoiningYear;
```

JoiningYear ▲ 1	NumberOfMembers
2015	1
2020	1
2021	1
2023	1
2024	1

16. SQL Stored Procedure

3) Write a stored procedure to retrieve all books by a particular author.

book_id	title	author	publisher	year_of_publication	price	genre
1	Making India Awesome	Chetan Bhagat	studio	2021	500	*NULL*
2	one indian girl	chetan n bhagat	first studio	1991	800	Love story
3	Revolution 2020	Balwant Gargi	second studio	1995	11550	Fiction

```
DELIMITER $$
CREATE PROCEDURE book()
BEGIN
  SELECT title FROM books WHERE author = "Chetan Bhagat";
```

END;

```
CALL book();
```

[Edit inline] [Edit] [Create P

Show all | Number

Extra options

title
Making India Awesome

4) Write a stored procedure that takes book_id as an argument and returns the price of the book.

DELIMITER $$

CREATE PROCEDURE Price()

BEGIN

SELECT price FROM Books WHERE book_id = '2';

END;

book_id	title	author	publisher	year_of_publication	price	genre
1	Making India Awesome	Chetan Bhagat	studio	2021	500	*NULL*
2	one indian girl	chetan n bhagat	first studio	1991	800	Love story
3	Revolution 2020	Balwant Gargi	second studio	1995	11550	Fiction

```
CALL Price();
```

[Edit inline] [Edit] [C

Show all | N

Extra options

price
800

17. SQL View

3) Create a view to show only the title, author, and price of books from the books table.

```
CREATE VIEW BookDetails AS SELECT title, author, price FROM books;
```

```
SELECT * FROM BookDetails;
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 Filter rows: Search this table

Extra options

	title	author	price
Edit Copy Delete	Making India Awesome	Chetan Bhagat	500
Edit Copy Delete	one indian girl	chetan n bhagat	800
Edit Copy Delete	Revolution 2020	Balwant Gargi	11550

4) Create a view to display members who joined before 2020.

```
CREATE VIEW MemberJoinedBefore2020 AS SELECT * FROM Members WHERE date_of_membership < '2020';
SELECT * FROM MemberJoinedBefore2020;
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 Filter rows: Search this table

Extra options

	member_id	member_name	date_of_membership	email
Edit Copy Delete	6	jack	2015	jacl01@gmail.com

18. SQL Trigger

3) Create a trigger to automatically update the last_modified timestamp of the books table whenever a record is updated.

```
CREATE TABLE triggertable(tbook_id int, ttitle varchar(80), tauthor varchar(80), tpublisher varchar(90), tyear_of_publication int, tprice int, tgenre varchar(80), ttime timestamp, action_perform varchar(80) );
```

```
ALTER TABLE books ADD COLUMN last_modified DATETIME;
```

```
CREATE TRIGGER update_books_last_modified BEFORE UPDATE ON books FOR EACH ROW BEGIN SET NEW.last_modified = CURRENT_TIMESTAMP; END;;
```

book_id	title	author	publisher	year_of_publication	price	genre
1	Making India Awesome	Chetan Bhagat	studio	2021	500	NULL
2	one indian girl	chetan n bhagat	first studio	1991	800	Love story
3	Revolution 2020	Balwant Gargi	second studio	1995	11550	Fiction
5	Book One	Author A	NULL	2021	NULL	NULL
6	Book Two	Author B	NULL	2022	NULL	NULL
7	Book Three	Author C	NULL	2023	NULL	NULL
8	Book Four	Author D	NULL	2024	NULL	NULL
101	The First Book	Author X	NULL	2021	NULL	NULL
102	The Second Book	Author Y	NULL	2022	NULL	NULL
103	The Third Book	Author Z	NULL	2023	NULL	NULL
104	The Fourth Book	Author A	NULL	2024	NULL	NULL

```
UPDATE books SET title = 'Me' WHERE book_id = 8;
```

```
SELECT * FROM books WHERE book_id = 8;
```

book_id	title	author	publisher	year_of_publication	price	genre	last_modified
8	Me	Author D	NULL	2024	NULL	NULL	2025-04-10 17:42:44

4) Create a trigger that inserts a log entry into a log_changes table whenever a DELETE operation is performed on the books table.

```
CREATE TABLE log_changes ( log_id INT PRIMARY KEY AUTO_INCREMENT, book_id INT, title VARCHAR(255), action VARCHAR(50), deleted_at DATETIME DEFAULT CURRENT_TIMESTAMP );
```

log_id	book_id	title	action	deleted_at

```
CREATE TRIGGER book_deletion AFTER DELETE ON books FOR EACH ROW BEGIN INSERT INTO log_changes (book_id, title, action) VALUES (OLD.book_id, OLD.title, 'Deleted'); END;;
```

book_id	title	author	publisher	year_of_publication	price	genre
1	Making India Awesome	Chetan Bhagat	studio	2021	500	NULL
2	one indian girl	chetan n bhagat	first studio	1991	800	Love story
3	Revolution 2020	Balwant Gargi	second studio	1995	11550	Fiction
5	Book One	Author A	NULL	2021	NULL	NULL
6	Book Two	Author B	NULL	2022	NULL	NULL
7	Book Three	Author C	NULL	2023	NULL	NULL
8	Book Four	Author D	NULL	2024	NULL	NULL
101	The First Book	Author X	NULL	2021	NULL	NULL
102	The Second Book	Author Y	NULL	2022	NULL	NULL
103	The Third Book	Author Z	NULL	2023	NULL	NULL
104	The Fourth Book	Author A	NULL	2024	NULL	NULL

log_id	book_id	title	action	deleted_at
1	104	The Fourth Book	Deleted	2025-04-10 17:29:14

19. Introduction to PL/SQL

3) Write a PL/SQL block to insert a new book into the books table and display a confirmation message.

```
BEGIN
 INSERT INTO books (book_id,title, author, genre) VALUES (9,"The Name of the Rose", "Umberto Eco", "Mystery");

 DBMS_OUTPUT.PUT_LINE('New book added successfully!');

END;
/
```

4) Write a PL/SQL block to display the total number of books in the books table.

```
DECLARE
 total_books NUMBER;
BEGIN
 SELECT COUNT(*) INTO total_books FROM books;
```

```
 DBMS_OUTPUT.PUT_LINE('Total number of books: ' || total_books);
END;
/
```

20. PL/SQL Syntax

3) Write a PL/SQL block to declare variables for book_id and price, assign values, and display the results.

```
DECLARE
 book_id NUMBER := 1234;
 price NUMBER := 19.99;
BEGIN
 DBMS_OUTPUT.PUT_LINE('Book ID: ' || book_id);
 DBMS_OUTPUT.PUT_LINE('Price: ' || price);
END;
/
```

4) Write a PL/SQL block using constants and perform arithmetic operations on book prices.

```
DECLARE

 book1_price CONSTANT NUMBER := 15.99;
 book2_price CONSTANT NUMBER := 24.95;

 total_price NUMBER;

BEGIN

 total_price := book1_price + book2_price;

 DBMS_OUTPUT.PUT_LINE('Book 1 Price: ' || book1_price);
 DBMS_OUTPUT.PUT_LINE('Book 2 Price: ' || book2_price);
 DBMS_OUTPUT.PUT_LINE('Total Price: ' || total_price);
END;
/
```

21. PL/SQL Control Structures

3) Write a PL/SQL block using IF-THEN-ELSE to check if a book's price is above $100and print a message accordingly.

```
DECLARE
 book_price NUMBER := 600;
BEGIN
 IF book_price > 500 THEN
  DBMS_OUTPUT.PUT_LINE('This book is expensive!');
 ELSE
  DBMS_OUTPUT.PUT_LINE('This book is reasonably priced.');
 END IF;
END;
/
```

4) Use a FOR LOOP in PL/SQL to display the details of all books one by one.

```
DECLARE
 total_books NUMBER;
BEGIN

 SELECT COUNT(*) INTO total_books FROM books;

 DBMS_OUTPUT.PUT_LINE('Total number of books: ' || total_books);
END;
/
```

22. SQL Cursors

3) Write a PL/SQL block using an explicit cursor to fetch and display all records from the members table.

```
DECLARE
 CURSOR member_cursor IS SELECT member_id,member_name, date_of_membership
FROM membersbackup;

 member_id_var NUMBER;
 member_name_var VARCHAR2(100);
 date_of_membership VARCHAR2(50);
BEGIN
 OPEN member_cursor;

 LOOP
  FETCH member_cursor INTO member_id_var, member_name_var,
date_of_membership_var;
  EXIT WHEN member_cursor%NOTFOUND;

  DBMS_OUTPUT.PUT_LINE('Member ID: ' || member_id_var);
  DBMS_OUTPUT.PUT_LINE('Member Name: ' || member_name_var);
  DBMS_OUTPUT.PUT_LINE('Membership Type: ' || membership_type_var);
  DBMS_OUTPUT.PUT_LINE('----------------------');
 END LOOP;

 CLOSE member_cursor;
END;
/
```

4) Create a cursor to retrieve books by a particular author and display their titles.

```
DECLARE
 author_name VARCHAR2(100) := 'J.K. Rowling';

 CURSOR book_cursor IS
  SELECT title FROM books WHERE author = author_name;
```

```
 book_title_var VARCHAR2(255);
BEGIN

 OPEN book_cursor;

 LOOP
  FETCH book_cursor INTO book_title_var;
  EXIT WHEN book_cursor%NOTFOUND;

  DBMS_OUTPUT.PUT_LINE('Book Title: ' || book_title_var);
 END LOOP;

 CLOSE book_cursor;
END;
/
```

23. Rollback and Commit Savepoint

3) Perform a transaction that includes inserting a new member, setting a SAVEPOINT, and rolling back to the savepoint after making updates.

```
START TRANSACTION;
INSERT INTO membersbackup (member_id, member_name, status) VALUES (101, 'Jacky ', 'active');
SAVEPOINT after_insertion;
UPDATE membersbackup SET status = 'inactive' WHERE member_id = 101;
UPDATE membersbackup SET member_name = 'Jane Doe' WHERE member_id = 101;
ROLLBACK TO after_insertion;
COMMIT;
```

member_id	member_name	date_of_membership	email	status
2	Dhruvin	2021	dp15@gmail.com	*NULL*
3	Zainil	2024	zp2@gmail.com	*NULL*
4	Shubham	2023	sp11@gmail.com	*NULL*
5	Rushi	2020	rp33@gmail.com	*NULL*
6	jack	2015	jacl01@gmail.com	*NULL*
101	Jacky	*NULL*	*NULL*	active

4) Use COMMIT after successfully inserting multiple books into the books table, then use ROLLBACK to undo a set of changes made after a savepoint.
START TRANSACTION;
INSERT INTO books (book_id, title, author, year_of_publication) VALUES
(101, 'The First Book', 'Author X', 2021),
(102, 'The Second Book', 'Author Y', 2022),
(103, 'The Third Book', 'Author Z', 2023);
COMMIT;
START TRANSACTION;
INSERT INTO books (book_id, title, author, year_of_publication) VALUES
(104, 'The Fourth Book', 'Author A', 2024)

SAVEPOINT before_updates;
UPDATE books SET author = 'author j' WHERE book_id = 102;
ROLLBACK TO before_updates;
COMMIT;

```
START TRANSACTION;
```

```
INSERT INTO books (book_id, title, author, year_of_publication) VALUES (101, 'The First Book', 'Author X', 2021), (102, 'The Second Book', 'Author Y', 2022), (103, 'The Third Book', 'Author Z', 2023);
```

```
COMMIT;
```

```
START TRANSACTION;
```

```
INSERT INTO books (book_id, title, author, year_of_publication) VALUES (104, 'The Fourth Book', 'Author A', 2024);
```

```
SAVEPOINT before_updates;
```

```
UPDATE books SET author = 'author j' WHERE book_id = 102;
```

```
ROLLBACK TO before_updates;
```

```
COMMIT;
```

book_id	title	author	publisher	year_of_publication	price	genre
1	Making India Awesome	Chetan Bhagat	studio	2021	500	NULL
2	one indian girl	chetan n bhagat	first studio	1991	800	Love story
3	Revolution 2020	Balwant Gargi	second studio	1995	11550	Fiction
5	Book One	Author A	NULL	2021	NULL	NULL
6	Book Two	Author B	NULL	2022	NULL	NULL
7	Book Three	Author C	NULL	2023	NULL	NULL
8	Book Four	Author D	NULL	2024	NULL	NULL
101	The First Book	Author X	NULL	2021	NULL	NULL
102	The Second Book	Author Y	NULL	2022	NULL	NULL
103	The Third Book	Author Z	NULL	2023	NULL	NULL
104	The Fourth Book	Author A	NULL	2024	NULL	NULL