**Name**:Akanksha Sabale

**MySQL Bookstore Project**

**-- Table structure for table `books`**

--

CREATE DATABASE project;

USE Project;

DROP TABLE IF EXISTS `books`;

CREATE TABLE `books` (

`id` int NOT NULL AUTO\_INCREMENT,

`url` varchar(255) DEFAULT NULL,

`title` text,

`upc` varchar(255) DEFAULT NULL,

`product\_type` varchar(255) DEFAULT NULL,

`price\_excl\_tax` decimal(10,0) DEFAULT NULL,

`price\_incl\_tax` decimal(10,0) DEFAULT NULL,

`tax` decimal(10,0) DEFAULT NULL,

`price` decimal(10,0) DEFAULT NULL,

`availability` int DEFAULT NULL,

`num\_reviews` int DEFAULT NULL,

`stars` int DEFAULT NULL,

`category` varchar(255) DEFAULT NULL,

`description` text,

PRIMARY KEY (`id`)

) ENGINE=InnoDB AUTO\_INCREMENT=999 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

-- The code to see Books table :

# 1. Data Retrieval and Filtering

**-- a) Find all books with price\_incl\_tax > $50**

use project;

select \*from books;

select category,price\_incl\_tax

from books



where price\_incl\_tax > 50;

**-- b) List all books in a specific category ,sorted by their price\_excl\_tax in descending order.**

select category,price\_excl\_tax

from books

where category ='science fiction'

order by price\_excl\_tax desc;

**-- c) A query to find books with 4 or more stars and an availability of 10 or less.**

select availability,stars

from books

where stars >= 4

and availability <=10;

# 2. Data Aggregation and Analysis

**-- a) A query to find the total number of books and the average price for each category.**

select category,count(\*) as total\_books,

avg(price\_incl\_tax) as avg\_price

from books

group by category;

**-- b)A query to identify the top 5 most-reviewed books.**

select id ,title ,num\_reviews,stars,category

from books

order by num\_reviews

limit 5;

**-- c)A query to calculate the average stars rating for books in each category.**

select category ,avg(stars) as average\_stars

from books

group by category;

# 3. Complex Queries and Reporting

**-- a) A query to create a report showing book titles, price\_incl\_tax, and a calculated**

**-- column named price\_level with values like &#39;Expensive&#39; (if price &gt; 40), &#39;Moderate&#39;**

**-- (if price between 20 and 40), and &#39;Affordable&#39; (if price &lt; 20).**

select title,price\_incl\_tax,

case

when price\_incl\_tax > 40 then 'expensive'

when price\_incl\_tax between 20 and 40 then 'moderate'

else 'affordable'

end as price\_level

from books

order by price\_incl\_tax;

**-- b)A query to find the book with the highest price within each category.**

select b.category, b.title, b.price\_incl\_tax

from books b

join(select category,max(price\_incl\_tax)as max\_price

from books

group by category)

m on b.category=m.category

and b.price\_incl\_tax =m.max\_price

order by b.category;

**-- c) A query to calculate the percentage of books in each category out of the total number of books.**

SELECT category,

COUNT(\*) \* 100 / (SELECT COUNT(\*) FROM books) AS percentage

FROM books

GROUP BY category;

SELECT title, category, price\_incl\_tax

FROM books

WHERE price\_incl\_tax = (

SELECT MAX(price\_incl\_tax)

FROM books b2

WHERE b2.category = books.category

);

SELECT title, price\_incl\_tax,

CASE

WHEN price\_incl\_tax > 40 THEN 'Expensive'

WHEN price\_incl\_tax BETWEEN 20 AND 40 THEN 'Moderate'

ELSE 'Affordable'

END AS price\_level

FROM books;

# 4. Data Maintenance and Updates.

**-- a) An UPDATE statement to change the tax and price\_incl\_tax for a specific book.**

update books

set tax=5.00,

price\_incl\_tax=120.0

where id=101;

**-- b) A DELETE statement to remove a book that is no longer in stock (availability = 0).**

SET SQL\_SAFE\_UPDATES = 0;

DELETE FROM books

WHERE availability = 0;

**#5. Database Design and Schema Improvements**

**-- A brief explanation of the proposed changes (e.g., adding a publisher tabl and a foreign key) and a CREATE TABLE statement for the new schema.**

**-- Publisher Table**

CREATE TABLE publisher (

publisher\_id INT PRIMARY KEY AUTO\_INCREMENT,

publisher\_name VARCHAR(100) NOT NULL

);

**-- Categories Table**

CREATE TABLE categories (

category\_id INT PRIMARY KEY AUTO\_INCREMENT,

category\_name VARCHAR(100) NOT NULL

);

**-- Books Table (with Foreign Keys)**

DROP TABLE books;

CREATE TABLE books (

id INT PRIMARY KEY AUTO\_INCREMENT,

title VARCHAR(150) NOT NULL,

category\_id INT,

publisher\_id INT,

price\_excl\_tax DECIMAL(10,2),

tax DECIMAL(10,2),

price\_incl\_tax DECIMAL(10,2),

stars DECIMAL(2,1),

availability INT,

num\_reviews INT,

FOREIGN KEY (category\_id) REFERENCES categories(category\_id),

FOREIGN KEY (publisher\_id) REFERENCES publisher(publisher\_id)

);