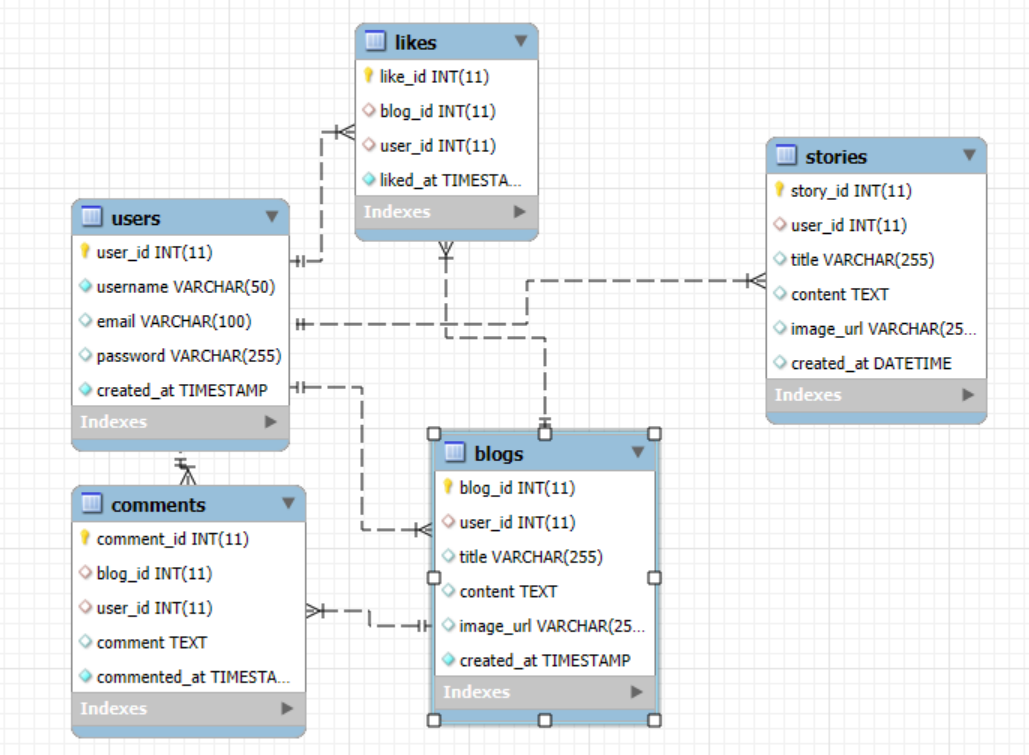
**UrbanShadows (SQL Case Study)**



**Create tables for the following table names**

|  |  |
| --- | --- |
| **-- 1. Users table**  CREATE TABLE users (  user\_id INT PRIMARY KEY AUTO\_INCREMENT,  username VARCHAR(50) NOT NULL UNIQUE,  email VARCHAR(100),  password VARCHAR(255),  created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP  ); | **-- 2. Blogs table**  CREATE TABLE blogs (  blog\_id INT PRIMARY KEY AUTO\_INCREMENT,  user\_id INT,  title VARCHAR(255),  content TEXT,  image\_url VARCHAR(255),  created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,  FOREIGN KEY (user\_id) REFERENCES users(user\_id)  ); |
| **-- 3. Likes table**  CREATE TABLE likes (  like\_id INT PRIMARY KEY AUTO\_INCREMENT,  blog\_id INT,  user\_id INT,  liked\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,  FOREIGN KEY (blog\_id) REFERENCES blogs(blog\_id),  FOREIGN KEY (user\_id) REFERENCES users(user\_id)  ); | **-- 4. Comments table**  CREATE TABLE comments (  comment\_id INT PRIMARY KEY AUTO\_INCREMENT,  blog\_id INT,  user\_id INT,  comment TEXT,  commented\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,  FOREIGN KEY (blog\_id) REFERENCES blogs(blog\_id),  FOREIGN KEY (user\_id) REFERENCES users(user\_id)  ); |
| **-- 5. Stories table**  CREATE TABLE stories (  story\_id INT PRIMARY KEY AUTO\_INCREMENT,  user\_id INT,  title VARCHAR(255),  content TEXT,  image\_url VARCHAR(255),  created\_at DATETIME DEFAULT CURRENT\_TIMESTAMP,  FOREIGN KEY (user\_id) REFERENCES users(user\_id)  ); |

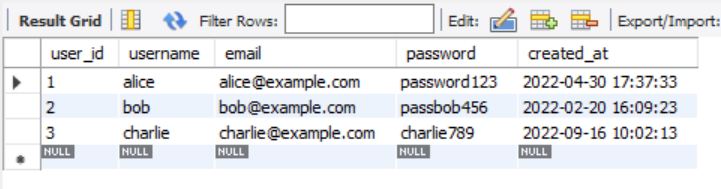
**Insert records into the following tables**

|  |
| --- |
| **-- Users Table**  INSERT INTO users (username, email, password, created\_at) VALUES  ('alice', 'alice@example.com', 'password123', '2022-04-30 17:37:33'),  ('bob', 'bob@example.com', 'passbob456', '2022-02-20 16:09:23'),  ('charlie', 'charlie@example.com', 'charlie789', '2022-09-16 10:02:13'); |
| **-- Blogs Table**  INSERT INTO blogs (user\_id, title, content, image\_url, created\_at) VALUES  (1, 'Urban Legends in India', 'Content about urban legends.', 'img1.jpg', '2022-08-30 12:58:02'),  (2, 'My Travel Story', 'Travel experience details.', 'img2.jpg', '2023-09-28 13:57:58'),  (3, 'Haunted Places', 'Story about haunted locations.', 'img3.jpg', '2022-10-03 04:40:50'); |
| **-- Likes Table**  INSERT INTO likes (blog\_id, user\_id, liked\_at) VALUES  (1, 2, '2023-10-08 16:28:39'),  (2, 3, '2022-01-26 09:20:38'),  (3, 1, '2023-06-23 12:40:52'); |
| **-- Comments Table**  INSERT INTO comments (blog\_id, user\_id, comment, commented\_at) VALUES  (1, 2, 'Great blog!', '2022-06-19 19:30:17'),  (2, 3, 'Nice story!', '2022-08-15 06:07:03'),  (3, 1, 'Creepy but good!', '2022-04-23 19:44:38'); |
| **-- Stories Table**  INSERT INTO stories (user\_id, title, content, image\_url, created\_at) VALUES  (1, 'Story of the Old Fort', 'There was an eerie silence in the air...', 'story1.jpg', '2022-05-12 10:52:48'),  (2, 'Mysterious Hilltop', 'Legends say lights are seen here...', 'story2.jpg', '2022-03-11 16:01:39'),  (3, 'Midnight Train', 'The train whistled with no one on board...', 'story3.jpg', '2022-05-20 05:04:33'); |

**Questions based on above mention tables and data**

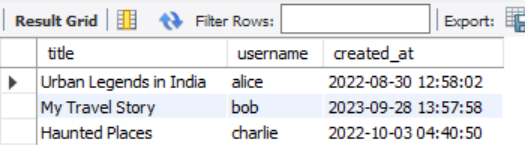
**Q1. Write an SQL query to retrieve all records from the users table, including all columns and all users.**

|  |
| --- |
| SELECT \* FROM users; |

****

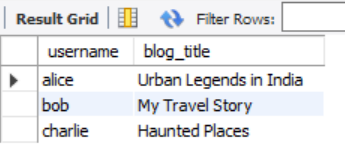
**Q2.** **Write an SQL query to list all blogs along with the corresponding user names who posted them. Assume there is a blogs table and a user’s table with a foreign key relationship**

|  |
| --- |
| SELECT b.title, u.username, b.created\_at FROM blogs b  INNER JOIN users u ON b.user\_id = u. user\_id; |

****

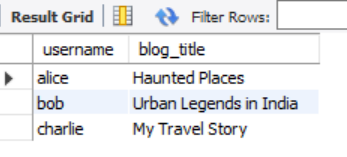
**Q3.** **Write an SQL query to list all users and their blogs, if any. Include users even if they have not written any blogs.**

|  |
| --- |
| SELECT u.username, b.title AS blog\_title FROM users u  LEFT JOIN blogs b ON u. user\_id = b.user\_id; |

****

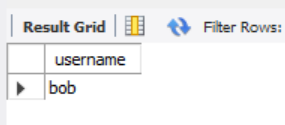
**Q4.** **Write an SQL query to retrieve the names of users who liked blogs, along with the titles of the blogs they liked.**

|  |
| --- |
| SELECT u.username, b.title AS blog\_title FROM likes l  RIGHT JOIN blogs b ON l.blog\_id = b. blog\_id  RIGHT JOIN users u ON l.user\_id = u. user\_id; |

****

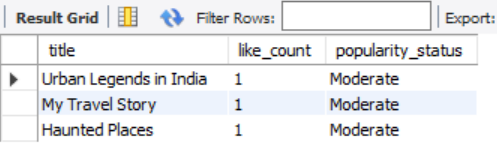
**Q5. Write an SQL query to retrieve the names of users who have commented on a specific blog, identified by its title.**

|  |
| --- |
| SELECT username FROM users  WHERE user\_id IN (  SELECT user\_id FROM comments  WHERE blog\_id = (SELECT blog\_id FROM blogs WHERE title = 'Urban Legends in India')  ); |

****

**Q6.** **Write an SQL query to show the popularity of each blog based on the number of likes it has received. Include the blog title and the total number of likes.**

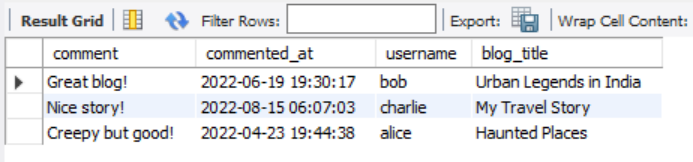
|  |
| --- |
| SELECT b.title, COUNT(l. like\_id) AS like\_count, CASE  WHEN COUNT(l. like\_id) >= 2 THEN 'Popular'  WHEN COUNT(l. like\_id) = 1 THEN 'Moderate'  ELSE 'Unnoticed'  END AS popularity\_status  FROM blogs b  LEFT JOIN likes l ON b.id = l.blog\_id  GROUP BY b. blog\_id; |

****

**Q7. Write an SQL statement to create a view that displays all comments along with the corresponding blog titles and the usernames of the commenters.**

|  |
| --- |
| CREATE VIEW comment\_details AS  SELECT c.comment, c.commented\_at, u.username, b.title AS blog\_title  FROM comments c  JOIN users u ON c.user\_id = u user\_id  JOIN blogs b ON c.blog\_id = b.blog\_id; |

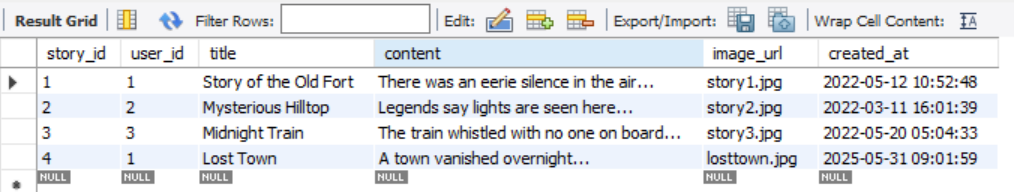
|  |
| --- |
| SELECT \* FROM comment\_details; |

****

**Q8.** **Write a stored procedure to insert a new story into the stories table. The procedure should accept parameters such as title, content, user ID, and date.**

|  |
| --- |
| DELIMITER $  CREATE PROCEDURE AddStory (  IN p\_user\_id INT,  IN p\_title VARCHAR(255),  IN p\_content TEXT,  IN p\_image\_url VARCHAR(255)  )  BEGIN  INSERT INTO stories (user\_id, title, content, image\_url, created\_at)  VALUES (p\_user\_id, p\_title, p\_content, p\_image\_url, NOW());  END $  DELIMITER ; |

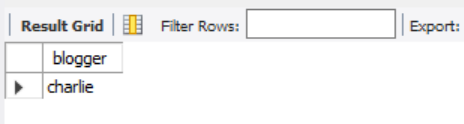
|  |
| --- |
| CALL AddStory(1, 'Lost Town', 'A town vanished overnight...', 'losttown.jpg'); |

****

**Q9.** **Write a cursor in SQL to iterate through all users who have posted blogs and display their usernames one by one. Simulate the logic using appropriate cursor declarations and control flow.**

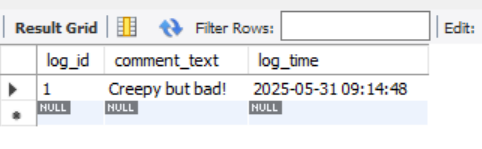
|  |
| --- |
| DELIMITER $  CREATE PROCEDURE ShowBloggers()  BEGIN  DECLARE done INT DEFAULT FALSE;  DECLARE v\_name VARCHAR(255);  DECLARE cur CURSOR FOR  SELECT DISTINCT u.username  FROM users u  JOIN blogs b ON u.user\_id = b.user\_id;  DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;  OPEN cur;  read\_loop: LOOP  FETCH cur INTO v\_name;  IF done THEN  LEAVE read\_loop;  END IF;  SELECT v\_name AS blogger;  END LOOP;  CLOSE cur;  END $  DELIMITER ; |

|  |
| --- |
| CALL ShowBloggers(); |

****

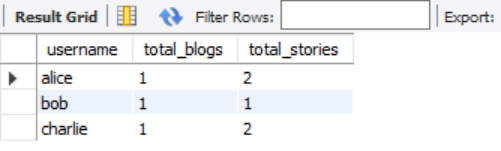
**Q10.** **Write a trigger that logs an entry whenever a new comment is added to the comments table. The log should record details such as the comment ID, user ID, blog ID, and timestamp.**

|  |
| --- |
| CREATE TABLE comment\_log (  log\_id INT AUTO\_INCREMENT PRIMARY KEY,  comment\_text TEXT,  log\_time DATETIME  );  DELIMITER $  CREATE TRIGGER after\_comment\_insert  AFTER INSERT ON comments  FOR EACH ROW  BEGIN  INSERT INTO comment\_log (comment\_text, log\_time)  VALUES (NEW.comment, NOW());  END $  DELIMITER ; |

****

**Q.11. Write an SQL query using aggregate functions and GROUP BY to show the total number of blogs and stories posted by each user.**

|  |
| --- |
| SELECT u.username,  COUNT(DISTINCT b.blog\_id) AS total\_blogs,  COUNT(DISTINCT s.story\_id) AS total\_stories  FROM users u  LEFT JOIN blogs b ON u.user\_id = b.user\_id  LEFT JOIN stories s ON u.user\_id = s.user\_id  GROUP BY u.user\_id; |

****