UrbanShadows SQL Queries - Notes Format

Q1. Retrieve All Users

SELECT * FROM users;

- SELECT *: Fetches all columns.

- FROM users: From the users table.

- Use: See all user records at once.

UrbanShadows SQL Queries - Notes Format

Q2. Blogs with Usernames

```
SELECT b.title, u.username, b.created_at
FROM blogs b
INNER JOIN users u ON b.user_id = u.user_id;
```

- INNER JOIN: Returns blogs with matching users only.
- Shows blog title & username.
- Use: Who posted which blog and when.

UrbanShadows SQL Queries - Notes Format

Q3. Users and Their Blogs (Even If None)

```
SELECT u.username, b.title AS blog_title
FROM users u
LEFT JOIN blogs b ON u.user_id = b.user_id;
```

- LEFT JOIN: Includes all users, even those with no blogs.
- Use: Match users to blogs or NULL if none.

UrbanShadows SQL Queries - Notes Format

Q4. Users Who Liked Blogs

```
SELECT u.username, b.title AS blog_title
FROM likes l
JOIN blogs b ON l.blog_id = b.blog_id
JOIN users u ON l.user_id = u.user_id;
```

- Joins likes, blogs, and users.
- Use: Show which user liked which blog.

UrbanShadows SQL Queries - Notes Format

Q5. Users Who Commented on a Specific Blog

```
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')
);
```

- Uses nested subqueries.
- Use: List commenters on specific blog.

UrbanShadows SQL Queries - Notes Format

Q6. Blog Popularity Based on 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.blog_id = l.blog_id
GROUP BY b.blog_id;
```

- Categorizes blogs by like count.
- Use: Rank blog posts by engagement.

UrbanShadows SQL Queries - Notes Format

Q7. View for Comments + Blog Titles + Usernames

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;

- VIEW: Saved query as virtual table.

- Use: Simplifies comment queries.

UrbanShadows SQL Queries - Notes Format

Q8. Procedure to Add a New Story

```
DELIMITER $

CREATE PROCEDURE AddStory (...)

BEGIN

  INSERT INTO stories (...)

  VALUES (..., NOW());

END $

DELIMITER ;


CALL AddStory(...);
```

- Automates story insertion.

- Use: Insert new story in one step.

UrbanShadows SQL Queries - Notes Format

Q9. Cursor to List Bloggers One-by-One

```
DELIMITER $

CREATE PROCEDURE ShowBloggers() BEGIN ... END $

DELIMITER ;


CALL ShowBloggers();
```

- CURSOR: Loops through bloggers.

- Use: Display one blogger at a time (simulation).

UrbanShadows SQL Queries - Notes Format

Q10. Trigger to Log New Comments

```
CREATE TABLE comment_log (...);

DELIMITER $

CREATE TRIGGER after_comment_insert

AFTER INSERT ON comments

FOR EACH ROW

BEGIN

  INSERT INTO comment_log (...);

END $

DELIMITER ;
```

- Trigger: Auto log after comment insert.

- Use: Track comments for audit/log.

UrbanShadows SQL Queries - Notes Format

Q11. Total Blogs & Stories per 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;
```

- Use: Track how active each user is.