```
SET SEARCH_PATH TO "DAILYHUNT";
```

1. List out total number of logged users.

=> Purpose: This query calculates and retrieves the total number of logged users from the LoggedUser table.

```
SELECT COUNT(*) AS total_logged_users FROM LoggedUser;
```

2. Total revenue for administrator

=> Purpose: This query calculates the total revenue generated from advertisements, which is relevant information for the administrator.

```
SELECT SUM(AdvertisementCost) AS total_revenue FROM Advertisement;
```

3. Retrieve the top 5 most expensive advertisements.

=> Purpose: This query identifies the top 5 most expensive advertisements based on the AdvertisementCost attribute.

```
SELECT * FROM Advertisement
ORDER BY AdvertisementCost DESC
LIMIT 5;
```

4. Find Users who are blocked

=> Purpose: This query retrieves information about users who are blocked, based on their UBlockStatus attribute being true.

```
UPDATE LoggedUser
SET UBlockStatus = true
WHERE UID IN (3001, 3003, 3005);
SELECT UID, UName, Email
FROM LoggedUser
WHERE UBlockStatus = true;
```

5. Find the most common types of reports submitted

=> Purpose: This query identifies the most common types of reports submitted based on the RType attribute in the Report_1 table.

```
SELECT RType, COUNT(*) AS Frequency FROM Report_1
GROUP BY RType
ORDER BY Frequency DESC
LIMIT 5;
```

6. Retrieve the details of the oldest active advertisement that is still running:

=> Purpose: To retrieve details about the oldest active advertisement that is still running, which provides insights into the longevity and ongoing effectiveness of advertising campaigns. This information can be useful for monitoring ad campaigns and assessing their duration.

```
SELECT *FROM Advertisement
WHERE DID IS NOT NULL
ORDER BY AdID ASC
```

LIMIT 1;

7. Retrieve the details of all the users who have liked a particular post with PostID 6001.

=> Purpose: This query fetches information about users who have liked a specific post identified by PostID.

```
SELECT * FROM LoggedUser
WHERE UID IN (SELECT UID FROM Likes_1 WHERE PostID = 6001);
```

8. Retrieve all the posts made by publishers located in Mumbai.

=> Purpose: This query fetches all posts made by publishers who are located in Mumbai, based on the address information stored in the Publisher table.

```
SELECT * FROM PublisherPost
WHERE PublisherID IN (SELECT PublisherID FROM Publisher WHERE
Address LIKE '%Mumbai%');
```

9. Retrieve all the posts made by users who are followed by user with UID 3005.

=> Purpose: This query fetches all posts made by users who are followed by a specific user identified by UID.

```
SELECT * FROM UserPost
WHERE UID IN (SELECT UID2 FROM Follow WHERE UID1 = 3005);
```

10. Retrieve the top 5 users who have the highest number of followers.

=> Purpose: This query identifies the top 5 users who have the highest number of followers by counting the number of followers each user has in the Follow table.

```
SELECT UID, UName, (SELECT COUNT(*) FROM Follow
WHERE UID2 = LoggedUser.UID) AS FollowerCount FROM LoggedUser
ORDER BY FollowerCount  DESC
LIMIT 5;
```

11. Retrieve the top 5 publishers who have the highest number of followers.

=> Purpose: This query identifies the top 5 publishers who have the highest number of followers by counting the number of followers each publisher has in the Follow2 table.

```
SELECT PublisherID, COUNT(UID) AS FollowerCount FROM Follow2
GROUP BY PublisherID
ORDER BY FollowerCount DESC
LIMIT 5;
```

12. Find the number of posts made by each user in the last month:

=> Purpose: To find out how many posts each user has made within the last month. This helps in analyzing user activity over time and identifying the most active contributors during a specific period.

```
SELECT UID, COUNT(*) AS NumPostsFROM UserPost
WHERE PostTime >= CURRENT_DATE - INTERVAL '1 month'
GROUP BY UID
```

13. Retrieve the names of users who have liked posts in the 'Reviews' category.

=> Purpose: This query identifies the names of users who have liked posts categorized under 'Reviews' by joining the LoggedUser, Likes_1, and UserPost tables.

```
SELECT DISTINCT LoggedUser.UName FROM LoggedUser
JOIN Likes_1 ON LoggedUser.UID = Likes_1.UID
JOIN UserPost ON Likes_1.PostID = UserPost.PostID
WHERE UserPost.Category = 'Reviews';
```

14. Retrieve the details of users who have liked a post in the 'Technology' genre.

=> Purpose: This query identifies users who have both liked a post in the 'Technology' genre.

```
SELECT DISTINCT LoggedUser.* FROM LoggedUser
JOIN Likes_1 ON LoggedUser.UID = Likes_1.UID
JOIN UserPost ON Likes_1.PostID = UserPost.PostID
WHERE UserPost.Genre = 'Technology';
```

15. Retrieve the topmost active publisher based on the number of posts made in the last week:

=> Purpose: To identify the publisher who have been the most active in publishing content within the last week. This information can be valuable for assessing the publishing frequency and contribution of different publishers over a short timeframe.

```
SELECT PublisherID, COUNT(*) AS NumPosts FROM PublisherPost
WHERE PostTime >= CURRENT_DATE - INTERVAL '1 week'
GROUP BY PublisherID
ORDER BY NumPosts DESC
LIMIT 1;
```

16. Find out the distribution of specific articles published by the publisher across different categories.

=> Purpose: This query determines the distribution of articles published by a specific publisher ('India Today') across different categories.

```
SELECT PP.Category, COUNT(*) AS article_count FROM PublisherPost PP
JOIN Publisher P ON PP.PublisherID = P.PublisherID
WHERE P.PublisherName = 'India Today'
GROUP BY PP.Category
ORDER BY PP.Category;
```

17. Find news articles with the highest number of shares on social media platforms.

=> Purpose: This query identifies news articles with the highest number of shares on social media platforms.

```
SELECT UP.PostID, UP.Title, COUNT(*) AS num_shares FROM UserPost UP
JOIN Share_1 S1 ON UP.PostID = S1.PostID
GROUP BY UP.PostID, UP.Title
ORDER BY num_shares DESC
LIMIT 1;
```

18. list out the category at which a specific publisher's majority of posts are related,

=> Purpose: This query lists out the category where the majority of posts made by a specific publisher (in this case, 'The Economic Times') are related.

```
SELECT PP.Category, COUNT(*) AS post_count FROM PublisherPost PP
JOIN Publisher P ON PP.PublisherID = P.PublisherID
WHERE P.PublisherName = 'The Economic Times'
GROUP BY PP.Category
ORDER BY COUNT(*) DESC
LIMIT 1;
```

19. Most Popular Articles in a Week.

=> Purpose: This query identifies the most popular articles in the last week based on the number of views received within that time frame.

```
SELECT UserPost.PostID, UserPost.Title, COUNT(View_L_1.ViewTime) AS
ViewsCount FROM UserPost
JOIN View_L_1 ON UserPost.PostID = View_L_1.PostID
WHERE View_L_1.ViewTime >= CURRENT_DATE - INTERVAL '1 week'
GROUP BY UserPost.PostID, UserPost.Title
ORDER BY ViewsCount DESC
LIMIT 3;
```

20. Find the average number of likes per post in the 'Technology' genre:

=> Purpose: To determine the average number of likes received per post specifically within the 'Technology' genre. This helps to understand the level of engagement or popularity of technology-related posts among users.

```
SELECT AVG(likes_count) AS AvgLikesPerPost FROM
(
   SELECT COUNT(*) AS likes_count
   FROM Likes_1
   WHERE PostID IN (SELECT PostID FROM UserPost WHERE Genre =
'Technology')
   GROUP BY PostID
) AS LikesPerPost;
```

21. Retrieve the details of the most recent post made by each user:

=> Purpose: To fetch the details of the most recent post made by each user in the database. This is useful for displaying recent activity by users and providing insights into their latest contributions or interactions.

```
SELECT * FROM UserPost
WHERE (UID, PostTime) IN (
   SELECT UID, MAX(PostTime) AS LatestPostTime
   FROM UserPost
   GROUP BY UID
);
```

22. retrieve the name of the most followed publisher along with the number of followers

=> Purpose: This query identifies the most followed publisher along with the number of followers they have.

```
SELECT P.PublisherName, COUNT(*) AS num_followers FROM Follow2 F
JOIN Publisher P ON F.PublisherID = P.PublisherID
WHERE F.PublisherID = (
   SELECT PublisherID FROM Follow2
   GROUP BY PublisherID
   ORDER BY COUNT(*) DESC
   LIMIT 1
)
GROUP BY P.PublisherName;
```

23. Find the average number of shares per post in each category:

=> Purpose: To calculate the average number of shares received per post in each category. This allows for understanding the level of engagement and social interaction for posts across different categories.

```
SELECT Category, AVG(shares_count) AS AvgSharesPerPost FROM
(
   SELECT P.Category, COUNT(*) AS shares_count
   FROM Share_1 S1
   JOIN UserPost P ON S1.PostID = P.PostID
   GROUP BY P.Category, S1.PostID
) AS SharesPerPost
GROUP BY Category;
```

24. Retrieve all advertisements along with the publisher name, department name, and total views received:

=> Purpose: This query fetches details of advertisements along with information about the publisher, department, and total views received by joining multiple tables.

```
SELECT Advertisement.AdID, Publisher.PublisherName,
AdministrationDepartment.DName AS Department_Name,
COUNT(View_L_2.ViewTime) AS Total_Views_Received
FROM Advertisement
INNER JOIN Contains ON Advertisement.AdID = Contains.AdID
INNER JOIN PublisherPost ON Contains.PPostID = PublisherPost.PPostID
INNER JOIN Publisher ON PublisherPost.PublisherID =
Publisher.PublisherID
```

```
INNER JOIN AdministrationDepartment ON Advertisement.DID =
AdministrationDepartment.DID

LEFT JOIN View_L_2 ON PublisherPost.PPostID = View_L_2.PPostID

GROUP BY Advertisement.AdID, Publisher.PublisherName,
AdministrationDepartment.DName;
```