



Instagram User Analytics

Description

- Instagram user analysis is simple and small project to obtain information from database that is provided. In this work, user has to write SQL query to get data from this database.
- Data that has been collected by this process can be used by the teams across the business to launch new marketing campaigns, decide on app features, track the success of the app by measuring user engagement and improve the experience altogether with business growth.

Approach

- The data set is observed and studied to understand the different categories and parameters of the data provided.
- This data set is executed to draw useful insights on the business proceedings.

Tech-Stack Used

- MySQL workbench is used to run all the queries in this project.
- This software is very handy to manage the dataset and draw useful insights.

Insights

- Learnt about the use of Order by, Group by, In, Not in functions.
- Use of different joins such as inner join, left join, right join ect and when to use them.

Rewarding Most Loyal Users

Find the 5 oldest users of the Instagram from the database provided.

SQL :

select * from users order by created_at limit 5;

The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'SCHEMAS' tree with 'ig_clone' selected. The main editor shows a SQL query: `select * from users order by created_at limit 5;`. The 'Result Grid' at the bottom displays the results of the query, showing 5 rows of user data. The 'Output' pane at the bottom shows the execution log with the message '5 row(s) returned'.

SQL Query:

```
select * from users
order by created_at
limit 5;
```

Result Grid:

#	id	username	created_at
1	80	Darby_Herzog	2016-05-06 00:14:21
2	67	Emilio_Bernier52	2016-05-06 13:04:30
3	63	Elenor88	2016-05-08 01:30:41
4	95	Nicole71	2016-05-09 17:30:22
5	38	Jordyn.Jacobson2	2016-05-14 07:56:26

Output:

#	Time	Action	Message	Duration / Fetch
1	11:59:14	select * from users order by created_at limit 5	5 row(s) returned	0.140 sec / 0.000 sec

Remind Inactive Users to Start Posting

Find the users who have never posted a single photo on Instagram

select * from users

where id not in (select distinct user_id from photos);

The screenshot shows a database management interface with a left sidebar, a central query editor, and a right sidebar. The left sidebar displays a tree view of schemas, including 'bms_db1', 'ig_clone', 'mysqldatabase', 'sakila', 'sys', and 'world'. The 'ig_clone' schema is expanded, showing tables like 'comments', 'follows', 'likes', 'photo_tags', 'photos', 'tags', and 'users'. The central query editor contains the following SQL code:

```
88
89 • select * from users
90 where id not in (select distinct user_id from photos);
91
92 • select username,user_id from likes
```

Below the query editor is a 'Result Grid' showing the results of the query. The grid has three columns: 'id', 'username', and 'created_at'. The results are as follows:

id	username	created_at
7	Kassandra_Homenick	2016-12-12 06:50:08
14	Jadyn81	2017-02-06 23:29:16
21	Rocio33	2017-01-23 11:51:15
24	Maxwell.Halvorson	2017-04-18 02:32:44
25	Tierra.Trantow	2016-10-03 12:49:21
34	Pearl7	2016-07-08 21:42:01
36	Ollie_Ledner37	2016-08-04 15:42:20
41	Mckenna17	2016-07-17 17:25:45
45	David.Osinski47	2017-02-05 21:23:37
49	Morgan.Kassulke	2016-10-30 12:42:31
53	Linnea59	2017-02-07 07:49:34
54	Duane60	2016-12-21 04:43:38
57	Julien_Schmidt	2017-02-02 23:12:48
66	Mike.Auer39	2016-07-01 17:36:15
68	Franco_Keebler64	2016-11-13 20:09:27
71	Nia_Haag	2016-05-14 15:38:50
74	Hulda.Macejkovic	2017-01-25 17:17:28
75	Leslie67	2016-09-21 05:14:01
76	Janelle.Nikolaus81	2016-07-21 09:26:09
80	Darby_Herzog	2016-05-06 00:14:21
81	Esther.Zulauf61	2017-01-14 17:02:34
83	Bartholome.Bernhard	2016-11-06 02:31:23
89	Jessyca_West	2016-09-14 23:47:05
90	Esmeralda.Mraz57	2017-03-03 11:52:27
91	Bethany20	2016-06-03 23:31:53

The right sidebar contains a 'SQLAdditions' section with a 'Jump to' button and a message: 'Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.' Below this are buttons for 'Result Grid', 'Form Editor', 'Field Types', 'Query Stats', and 'Execution Plan'. At the bottom of the right sidebar are buttons for 'Apply', 'Revert', 'Context Help', and 'Snippets'.

Declaring Contest Winner

Identify the winner of the contest and provide their details to the team

select username,user_id from likes

right join users on users.id=likes.user_id

where(photo_id)=(select max(photo_id) from likes);

The screenshot shows a SQL IDE interface. On the left is a 'SCHEMAS' pane with a tree view containing databases like 'bms_db1', 'ig_clone', 'mysqlatabase', 'sakila', 'sys', and 'world'. The 'ig_clone' database is selected, showing tables like 'comments', 'follows', 'likes', 'photo_tags', 'photos', 'tags', and 'users'. Below the tree is a 'No object selected' message. The main area displays 'Query 1' with the following SQL code:

```
90 where id not in (select distinct user_id from photos);
91
92 • select username,user_id from likes
93 right join users on users.id=likes.user_id
94 where(photo_id)=(select max(photo_id) from likes);
95
```

Below the query editor is a 'Result Grid' showing the results of the query. The grid has two columns: 'username' and 'user_id'. The results are as follows:

username	user_id
Aiyana_Hoeger	31
Yvette.Gottlieb91	33
Lennie_Hartmann40	35
Ollie_Ledner37	36
Kelsi26	39
Mckenna17	41
Janet.Armstrong	43
Harrison.Beatty50	47
Granville_Kutch	48
Duane60	54
Julien_Schmidt	57
Sam52	60
Jayson65	61
Adelle96	65
Mike.Auer39	66
Erick5	70
Nia_Haag	71
Jaylan.Lakin	73
Leslie67	75
Janelle.Nikolaus81	76
Aracely.Johnston98	82
Bethany20	91
Frederik_Rice	92
Nicole71	95
Tomas.Beatty93	97

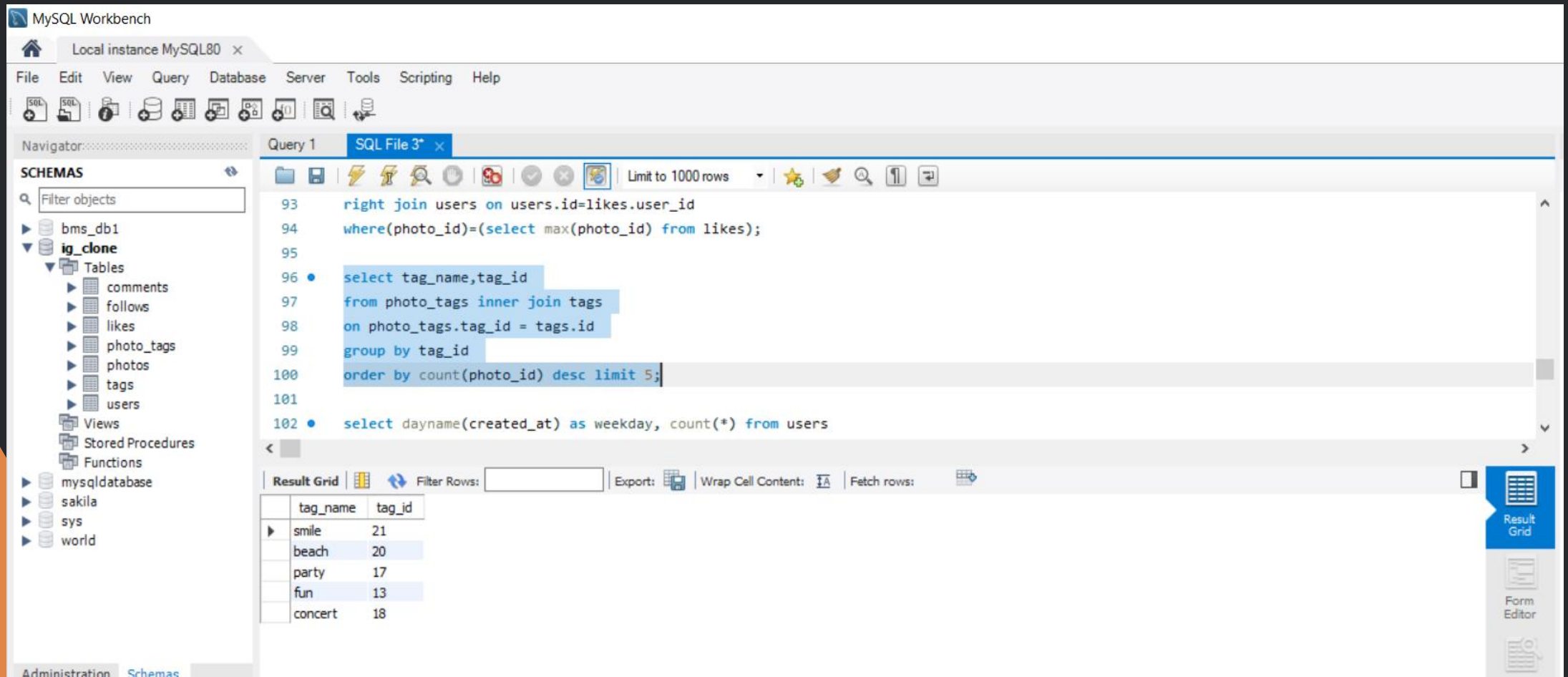
On the right side of the IDE, there is a vertical toolbar with icons for 'Result Grid', 'Form Editor', 'Field Types', 'Query Stats', and 'Execution Plan'.

Hashtag Researching

Identify and suggest the top 5 most commonly used hashtags on the platform.

```
select tag_name,tag_id from photo_tags inner join tags on photo_tags.tag_id = tags.id
```

```
group by tag_id order by count(photo_id) desc limit 5;
```



The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'SCHEMAS' tree with 'ig_clone' selected. The main editor window shows a SQL query in 'Query 1' with the following code:

```
93 right join users on users.id=likes.user_id
94 where(photo_id)=(select max(photo_id) from likes);
95
96 • select tag_name,tag_id
97   from photo_tags inner join tags
98   on photo_tags.tag_id = tags.id
99   group by tag_id
100  order by count(photo_id) desc limit 5;
101
102 • select dayname(created_at) as weekday, count(*) from users
```

The query results are displayed in the 'Result Grid' at the bottom, showing the top 5 most commonly used hashtags:

tag_name	tag_id
smile	21
beach	20
party	17
fun	13
concert	18

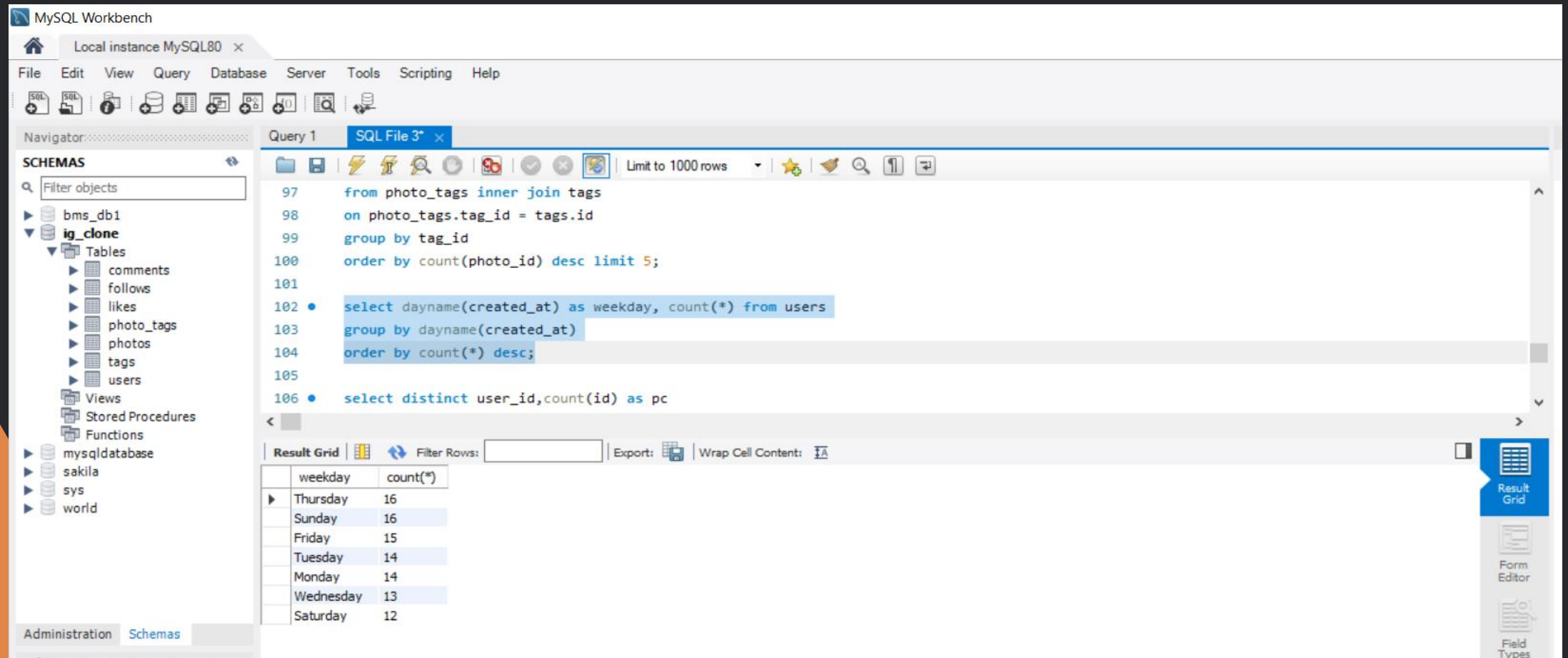
Launch AD Campaign

What day of the week do most users register on? Provide insights on when to schedule an ad campaign

```
select dayname(created_at) as weekday, count(*) from users
```

```
group by dayname(created_at)
```

```
order by count(*) desc;
```



The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'SCHEMAS' tree with 'ig_clone' selected, showing tables like 'comments', 'follows', 'likes', 'photo_tags', 'photos', 'tags', and 'users'. The main editor window shows a SQL query in 'Query 1' with line numbers 97 to 106. The query is:

```
97 from photo_tags inner join tags
98 on photo_tags.tag_id = tags.id
99 group by tag_id
100 order by count(photo_id) desc limit 5;
101
102 • select dayname(created_at) as weekday, count(*) from users
103 group by dayname(created_at)
104 order by count(*) desc;
105
106 • select distinct user_id, count(id) as pc
```

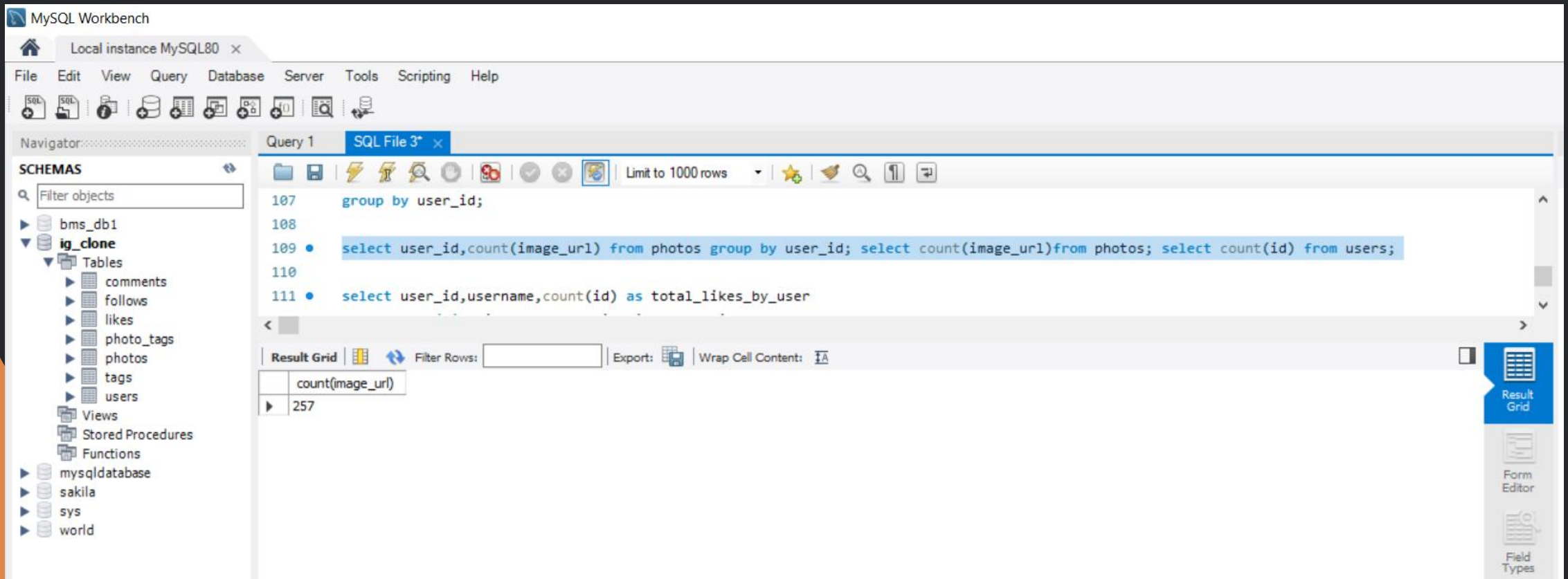
Below the query editor, the 'Result Grid' is displayed, showing the results of the query. The table has two columns: 'weekday' and 'count(*)'. The results are as follows:

weekday	count(*)
Thursday	16
Sunday	16
Friday	15
Tuesday	14
Monday	14
Wednesday	13
Saturday	12

User Engagement

Provide how many times does average user posts on Instagram. Also, provide the total number of photos on Instagram/total number of users

```
select user_id,count(image_url) from photos group by user_id; select count(image_url)from photos; select count(id) from users;
```



The screenshot displays the MySQL Workbench interface. On the left, the 'SCHEMAS' panel shows a tree view with databases like 'bms_db1', 'ig_clone', 'mysql', 'sakila', 'sys', and 'world'. The 'ig_clone' database is selected, showing tables such as 'comments', 'follows', 'likes', 'photo_tags', 'photos', 'tags', and 'users'. The main editor window, titled 'Query 1', contains the following SQL code:

```
107 group by user_id;
108
109 • select user_id,count(image_url) from photos group by user_id; select count(image_url)from photos; select count(id) from users;
110
111 • select user_id,username,count(id) as total_likes_by_user
```

Below the query editor, the 'Result Grid' tab is active, showing the results of the query. The first result is a single row with the value '257' under the column 'count(image_url)'.

count(image_url)
257

Bots & Fake Accounts

Provide data on users (bots) who have liked every single photo on the site (since any normal user would not be able to do this)

```
select user_id,username,count(id) as total_likes_by_user from users join likes on users.id=likes.user_id
```

```
group by user_id having total_likes_by_user = (select count(*) from photos);
```

The screenshot shows a SQL IDE interface. On the left is a 'Navigator' pane with a tree view of databases and tables. The 'ig_clone' database is selected, showing tables like 'comments', 'follows', 'likes', 'photo_tags', 'photos', 'tags', and 'users'. The main area displays a SQL query in 'Query 1' and its results in a 'Result Grid'.

Query 1:

```
108
109 • select user_id,count(image_url) from photos group by user_id; select count(image_url)from photos; select count(id) from users;
110
111 • select user_id,username,count(id) as total_likes_by_user
112 from users join likes on users.id=likes.user_id
113 group by user_id
114 having total_likes_by_user = (select count(*) from photos);
115
116
117
```


Result Grid:

	user_id	username	total_likes_by_user
▶	5	Aniya_Hackett	257
	14	Jadyn81	257
	21	Rocio33	257
	24	Maxwell.Halvorson	257
	36	Ollie_Ledner37	257
	41	Mckenna17	257
	54	Duane60	257
	57	Julien_Schmidt	257
	66	Mike.Auer39	257
	71	Nia_Haag	257
	75	Leslie67	257
	76	Janelle.Nikolaus81	257
	91	Bethany20	257

The interface also includes a 'Filter Rows' field, 'Export' and 'Wrap Cell Content' buttons, and a sidebar with 'Result Grid', 'Form Editor', 'Field Types', and 'Query Stats' options. The status bar at the bottom indicates 'Result 15' and 'Read Only'.

Result

Understood Joins and the different functions present in SQL and how data can be used to draw meaningful insights.





Thank You