



Programming Fundamentals of Analytics

CRN - 12794

Assignment one - MySQL

TOPIC: IG_CLONE_DATA

NAME: ADITYA SHROFF

COLLEGE ID: 101539669

DATE: 28/09/24



Q1. Store the 5 oldest users in a table

Limit to 1000 rows

```
2 • CREATE TABLE oldest_users AS
3   SELECT * FROM users
4   ORDER BY created_at
5   LIMIT 5;
6
7
8 • Select * From oldest_users;
9
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: [IA](#)

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



Q2. Store the two days with the most registrations

```
10
11 • CREATE TABLE top_registration_days AS
12 SELECT
13     DAYNAME(created_at) AS day,
14     COUNT(*) AS total
15 FROM users
16 GROUP BY day
17 ORDER BY total DESC
18 LIMIT 2;
19
20 • Select * from top_registration_days; |
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

	day	total
▶	Thursday	16
	Sunday	16

Q3 Store the usernames of users who have never uploaded a photo

```
21
22
23 • drop table no_photos_users;
24
25 • CREATE TABLE no_photos_users AS
26 SELECT username
27 FROM users
28 left JOIN photos
29     ON users.id = photos.user_id
30 WHERE photos.id IS NULL;
31
32 • Select * from no_photos_users;
33
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

username
Nia_Haag
Ollie_Ledner37
Pearl7
▶ Rocio33
Tierra.Trantow

no_photos_users x | Read Only | Context Help | Snippets

Output

#	Time	Action	Message	Duration / Fetch
✓ 48	20:29:45	Select * from no_photos_users LIMIT 0, 1000	26 row(s) returned	0.000 sec / 0.000 sec

Automatic context help disabled. Use the toolbar manually get help for the current caret position or toggle automatic help.



Q4. Store the photo with the highest number of likes

SQL File 3* netflix_mysql1 world_mysql SQL File 7* SQL File 6* Lucky_Shrub students_survey_mysql Create_Alter_Drop_Table* ig_clon

```
36 • CREATE TABLE most_liked_photo AS
37 SELECT
38     username,
39     photos.id,
40     photos.image_url,
41     COUNT(*) AS total
42 FROM photos
43 INNER JOIN likes
44     ON likes.photo_id = photos.id
45 INNER JOIN users
46     ON photos.user_id = users.id
47 GROUP BY photos.id
48 ORDER BY total DESC
49 LIMIT 1;
50
51 • Select * from most_liked_photo;
52
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

username	id	image_url	total
Zack_Kemmer93	145	https://jarret.name	48

Q5. Store the average number of photos per user

netflix_mysql1 world_mysql SQL File 7 SQL File 6 Lucky_Shrub students_survey_mysql Create_Alter_Drop_Table ig_clone_data SQL File 10 SQLA

```
53
54
55 • CREATE TABLE avg_photos_per_user AS
56 SELECT (SELECT Count(*)
57     FROM photos) / (SELECT Count(*)
58     FROM users) AS avg;
59
60 • Select * from avg_photos_per_user;
61
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

avg
2.5700

tos_per_user1 x Read Only Conte

Output

#	Time	Action	Message
1	03:00:27	CREATE TABLE avg_photos_per_user AS SELECT (SELECT Count(*) FROM photos) / (SELECT Count(*) FROM users) AS avg;	1 row(s) affected Records: 1 Duplicates: 0 Warnings: 0
2	03:01:11	CREATE TABLE avg_photos_per_user AS SELECT (SELECT Count(*) FROM photos) / (SELECT Count(*) FROM users) AS avg;	Error Code: 1050 Table 'avg_photos_per_user' already exists
3	03:01:27	Select * from avg_photos_per_user LIMIT 0, 1000	1 row(s) returned



Q6. Store the top 5 most popular tags used in photos

The screenshot shows a SQL IDE with the following SQL query:

```
60 • Select * from avg_photos_per_user;
61
62 • drop table most_popular_tags;
63
64 • Create table most_popular_tags AS
65   select photo_tags.photo_id
66   from photo_tags
67   join photos
68   ON photo_tags.photo_id = photos.id
69   group by photo_id
70   order by photo_id Desc
71   limit 5;
72
73 • select * from most_popular_tags;
```

The results grid shows the following data:

photo_id
257
254
253
251
250

The output pane shows the execution of the query:

```
1 03:00:27 CREATE TABLE avg_photos_per_user AS SELECT (SELECT Count(*) FROM photos) / (SELECT Count(*) FROM users) AS avg_likes;
1 row(s) affected Records: 1 Duplicates: 0 Warnings: 0 0.015 sec
```

Q7. Store the usernames of users who have given exactly as many likes as there are total photos

The screenshot shows a SQL IDE with the following SQL query:

```
75 • CREATE TABLE equal_likes_users AS
76   SELECT username,
77     Count(*) AS num_likes
78   FROM users
79   INNER JOIN likes
80   ON users.id = likes.user_id
81   GROUP BY likes.user_id
82   HAVING num_likes = (SELECT Count(*) FROM photos);
83
84
85 • SELECT * FROM oldest_users;
86 • SELECT * FROM top_registration_days;
87 • SELECT * FROM no_photos_users;
88 • SELECT * FROM most_liked_photo;
89 • SELECT * FROM avg_photos_per_user;
90 • SELECT * FROM top_tags;
91 • SELECT * FROM equal_likes_users;
92
93
```

The results grid shows the following data:

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