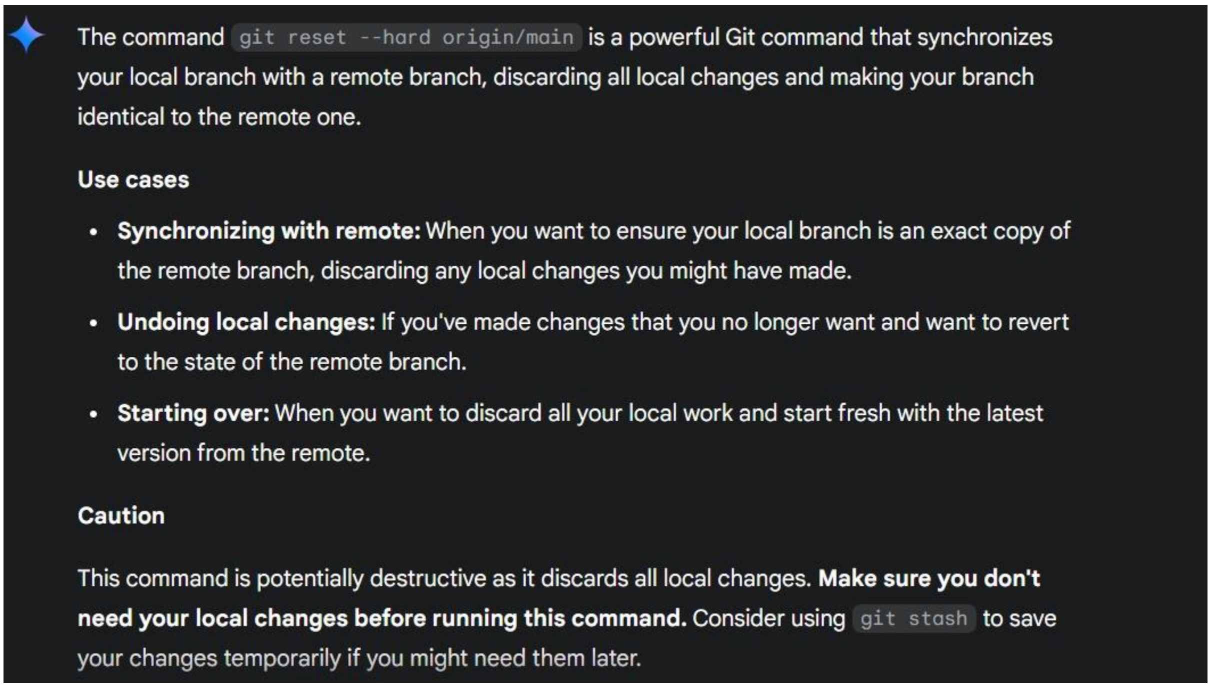
SRE day -4 (github)

Now to deal with the scenario where if pull request doesn't work, as our local is too much behind, then we use case a specific command which reset and overrides our local changes,

But before that do keep a backup of all the work that you have done on local machine after the original data in repository.



The command `git reset --hard origin/main` is a powerful Git command that synchronizes your local branch with a remote branch, discarding all local changes and making your branch identical to the remote one.

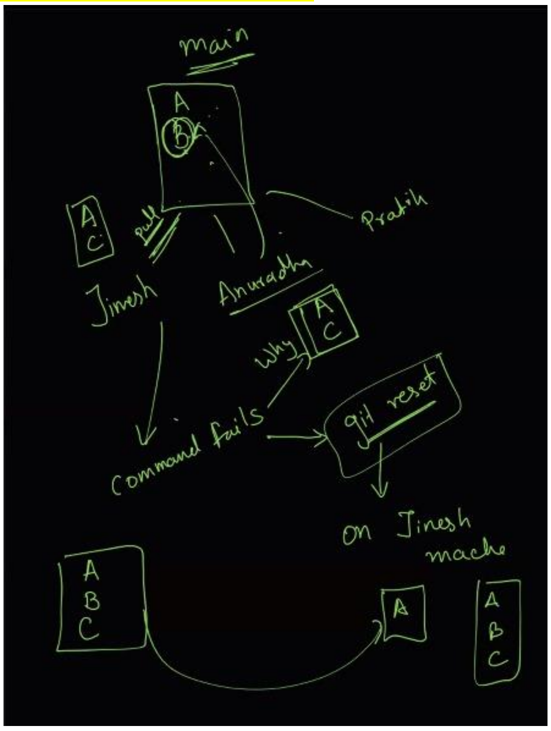
Use cases

- **Synchronizing with remote:** When you want to ensure your local branch is an exact copy of the remote branch, discarding any local changes you might have made.
- **Undoing local changes:** If you've made changes that you no longer want and want to revert to the state of the remote branch.
- **Starting over:** When you want to discard all your local work and start fresh with the latest version from the remote.

Caution

This command is potentially destructive as it discards all local changes. **Make sure you don't need your local changes before running this command.** Consider using `git stash` to save your changes temporarily if you might need them later.

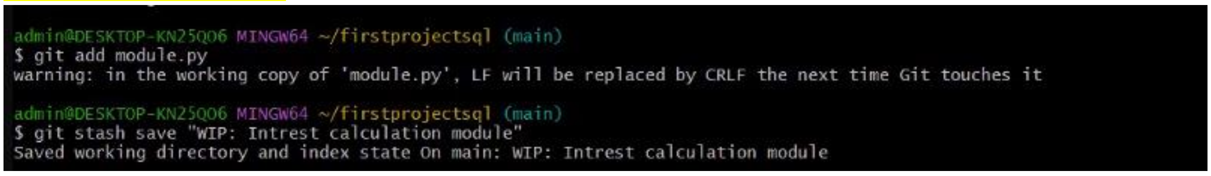
Example to understand this concept:



Now to avoid the situation of losing data , we use a phenomena named “**stashing**”,

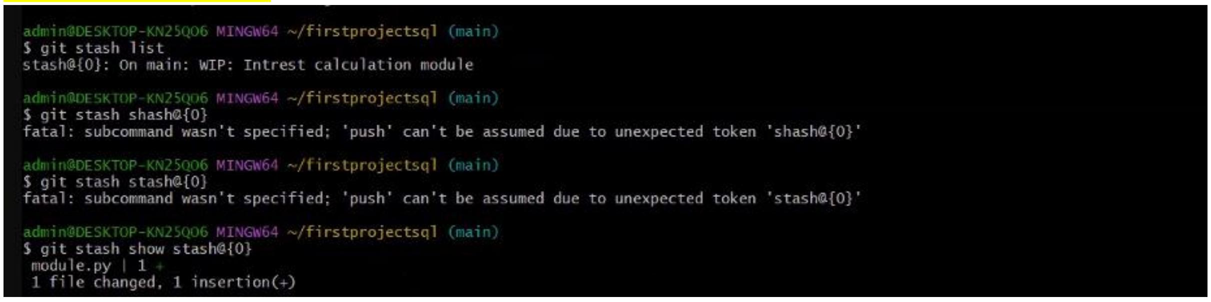
so to avoid it that our local machine is behind the original ui, we do stashing for temporary solution
or another simple way can be that we can just copy all our local data and then do hard reset and then get latest data on main branch and then merge our copied original code and paste it

How to stash and save info for storing our changes in temporary form:



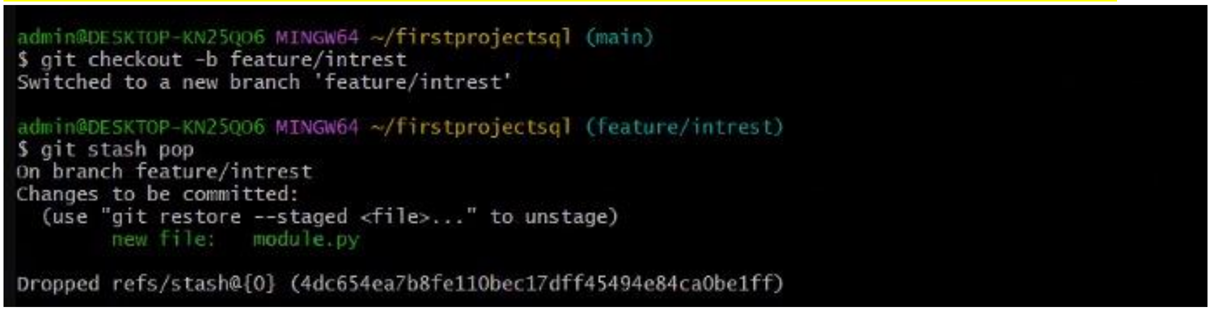
```
admin@DESKTOP-KN25Q06 MINGW64 ~/firstprojectsql (main)
$ git add module.py
warning: in the working copy of 'module.py', LF will be replaced by CRLF the next time Git touches it

admin@DESKTOP-KN25Q06 MINGW64 ~/firstprojectsql (main)
$ git stash save "WIP: Intrest calculation module"
Saved working directory and index state On main: WIP: Intrest calculation module
```

Now to show all the available stashes stored in our local machine (or hard drive):



```
admin@DESKTOP-KN25Q06 MINGW64 ~/firstprojectsql (main)
$ git stash list
stash@{0}: On main: WIP: Intrest calculation module

admin@DESKTOP-KN25Q06 MINGW64 ~/firstprojectsql (main)
$ git stash shash@{0}
fatal: subcommand wasn't specified; 'push' can't be assumed due to unexpected token 'shash@{0}'

admin@DESKTOP-KN25Q06 MINGW64 ~/firstprojectsql (main)
$ git stash stash@{0}
fatal: subcommand wasn't specified; 'push' can't be assumed due to unexpected token 'stash@{0}'

admin@DESKTOP-KN25Q06 MINGW64 ~/firstprojectsql (main)
$ git stash show stash@{0}
 module.py | 1 +
 1 file changed, 1 insertion(+)
```

Now to retrieve a specific stash into some specific branch then:

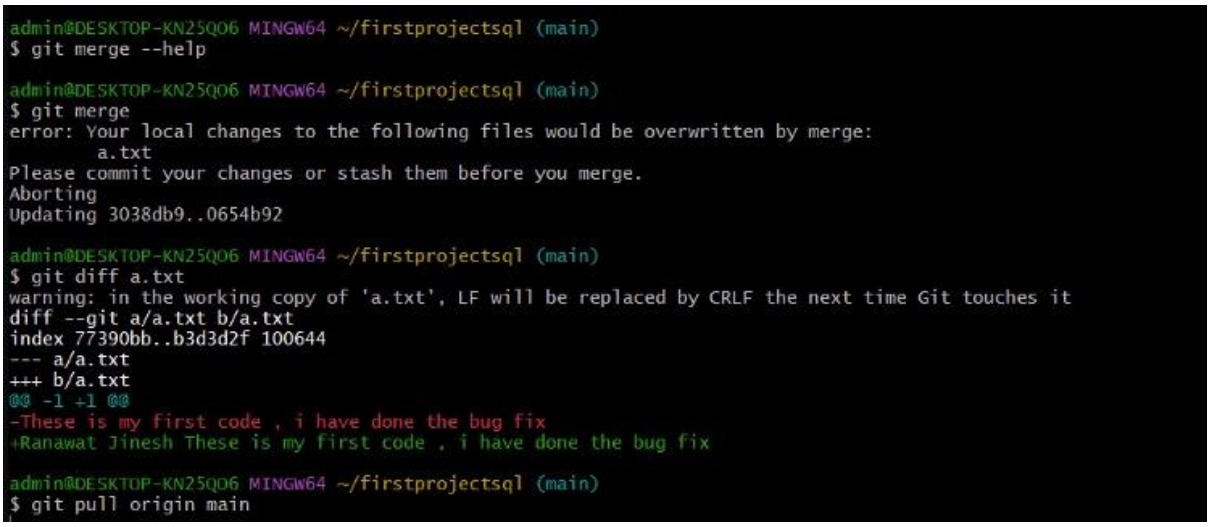


```
admin@DESKTOP-KN25Q06 MINGW64 ~/firstprojectsql (main)
$ git checkout -b feature/intrest
Switched to a new branch 'feature/intrest'

admin@DESKTOP-KN25Q06 MINGW64 ~/firstprojectsql (feature/intrest)
$ git stash pop
On branch feature/intrest
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   module.py

Dropped refs/stash@{0} (4dc654ea7b8fe110bec17dff45494e84ca0be1ff)
```

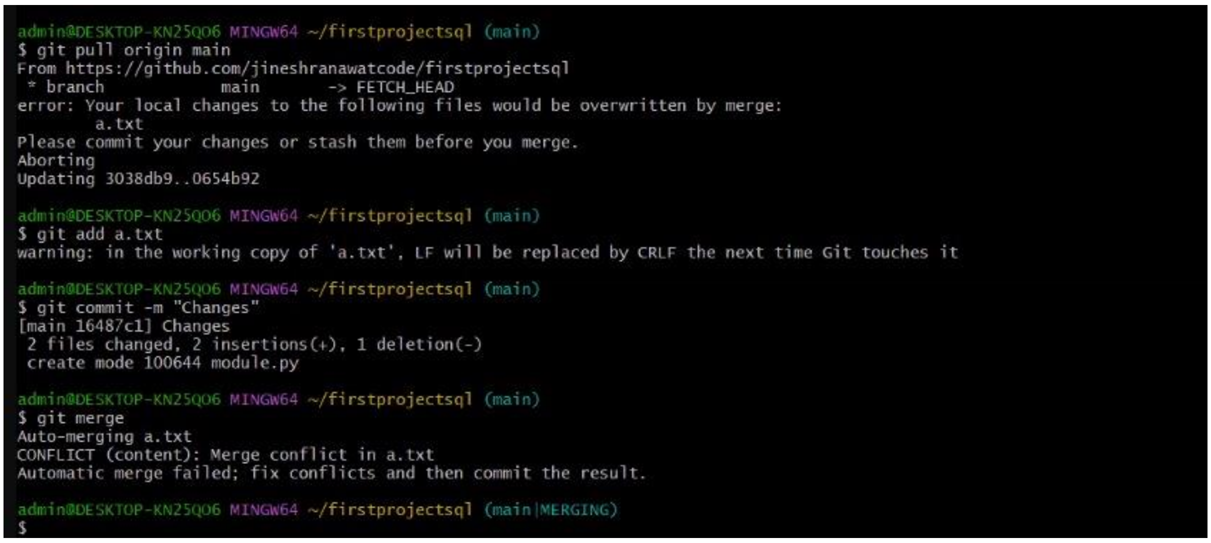
- git stash to add .
- git stash list to see all the lists of stashes .
- and then we change branch if we want to .
- and the we pop stash .

git stash usually stores these files on our hard drive, but if we want to secure it more then we can store that file in one drive so that it doesn't get corrupted.

Now we would see the command for pulling, it is used before every push it is used to bring our local machine data , up to date with reference to github repository data. And with it we would also see how we perform merge and compare from local machine:



```
admin@DESKTOP-KN25Q06 MINGW64 ~/firstprojectsql (main)
$ git merge --help

admin@DESKTOP-KN25Q06 MINGW64 ~/firstprojectsql (main)
$ git merge
error: Your local changes to the following files would be overwritten by merge:
        a.txt
Please commit your changes or stash them before you merge.
Aborting
Updating 3038db9..0654b92

admin@DESKTOP-KN25Q06 MINGW64 ~/firstprojectsql (main)
$ git diff a.txt
warning: in the working copy of 'a.txt', LF will be replaced by CRLF the next time Git touches it
diff --git a/a.txt b/a.txt
index 77390bb..b3d3d2f 100644
--- a/a.txt
+++ b/a.txt
@@ -1 +1 @@
-These is my first code , i have done the bug fix
+Ranawat Jinesh These is my first code , i have done the bug fix

admin@DESKTOP-KN25Q06 MINGW64 ~/firstprojectsql (main)
$ git pull origin main
```



```
admin@DESKTOP-KN25Q06 MINGW64 ~/firstprojectsql (main)
$ git pull origin main
From https://github.com/jineshranawatcode/firstprojectsql
 * branch            main       -> FETCH_HEAD
error: Your local changes to the following files would be overwritten by merge:
        a.txt
Please commit your changes or stash them before you merge.
Aborting
Updating 3038db9..0654b92

admin@DESKTOP-KN25Q06 MINGW64 ~/firstprojectsql (main)
$ git add a.txt
warning: in the working copy of 'a.txt', LF will be replaced by CRLF the next time Git touches it

admin@DESKTOP-KN25Q06 MINGW64 ~/firstprojectsql (main)
$ git commit -m "Changes"
[main 16487c1] Changes
 2 files changed, 2 insertions(+), 1 deletion(-)
 create mode 100644 module.py

admin@DESKTOP-KN25Q06 MINGW64 ~/firstprojectsql (main)
$ git merge
Auto-merging a.txt
CONFLICT (content): Merge conflict in a.txt
Automatic merge failed; fix conflicts and then commit the result.

admin@DESKTOP-KN25Q06 MINGW64 ~/firstprojectsql (main|MERGING)
$
```

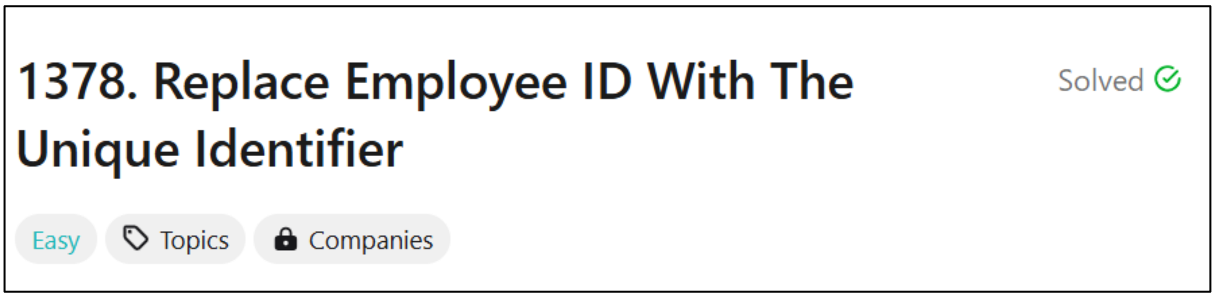
Now if we notice when we try to "merge" it gives rise to a merge conflict, **as we need to manually open the file and remove extra keywords from our file.** (now these keywords generally are MAIN and Remote)

here earlier we were manually changing the data, but now we would simply use merge command after pull and it would merge the files from git with our new file and after this we need to change manually and remove some keywords and then we can simply add the file and then commit and push.

LIST OF COMMANDS USED FOR ABOVE MENTION OPERATIONS:

- git checkout feature/intrest
- vi a.txt
- git checkout main
- git merge
- git merge feature/intrest
- ls -lrt
- vi a.txt
- git merge --help
- git merge
- diff a.txt
- git pull origin main
- git add a.txt
- git commit -m "Changes"
- git merge
- pwd
- git add a.txt
- git commit -m "Changes"
- git push -u origin main
- ls -lrt
- vi module.py
- git add module.py
- git commit -m "Changes adding Jinesh"
- git push -u origin main
- git pull
- merge
- add module.py
- commit -m "Changes"
- push -u origin main

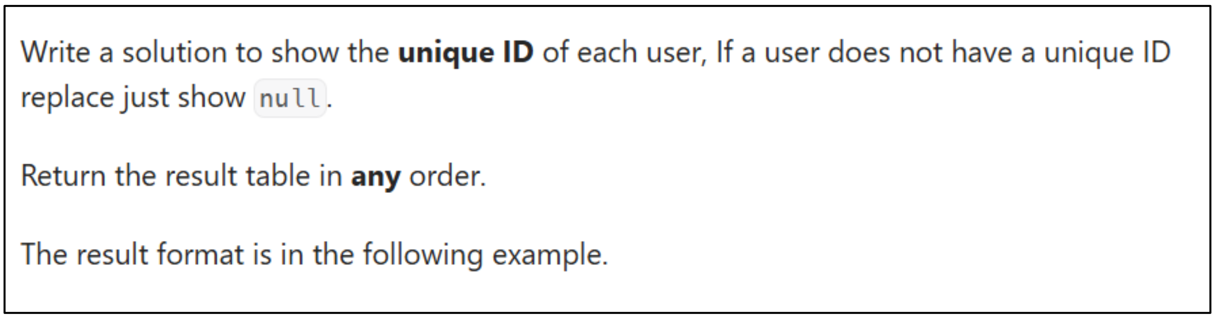
Leet-code important SQL Questions:



1378. Replace Employee ID With The Unique Identifier

Solved ✓

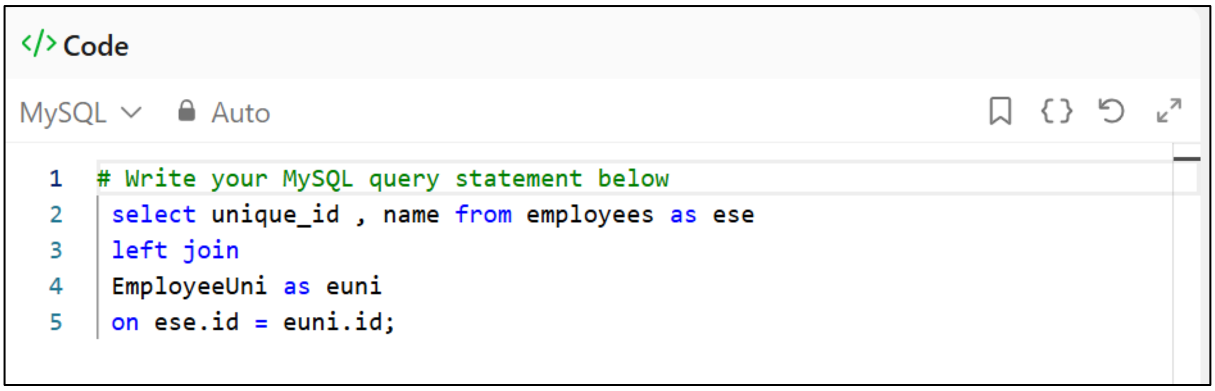
Easy Topics Companies



Write a solution to show the **unique ID** of each user, If a user does not have a unique ID replace just show `null`.

Return the result table in **any** order.

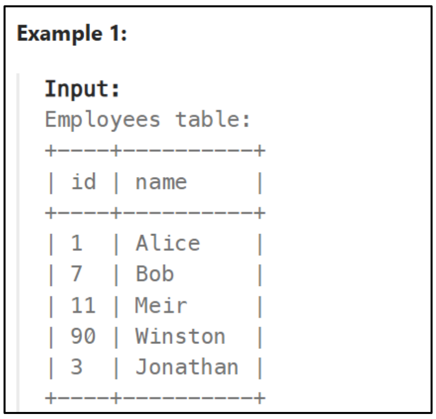
The result format is in the following example.



</> Code

MySQL Auto

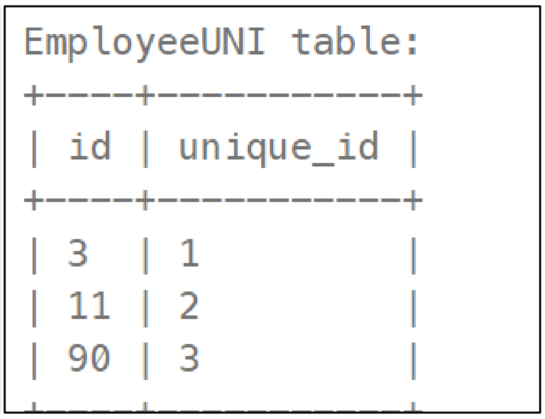
```sql
# Write your MySQL query statement below
select unique_id , name from employees as ese
left join
EmployeeUni as euni
on ese.id = euni.id;
```



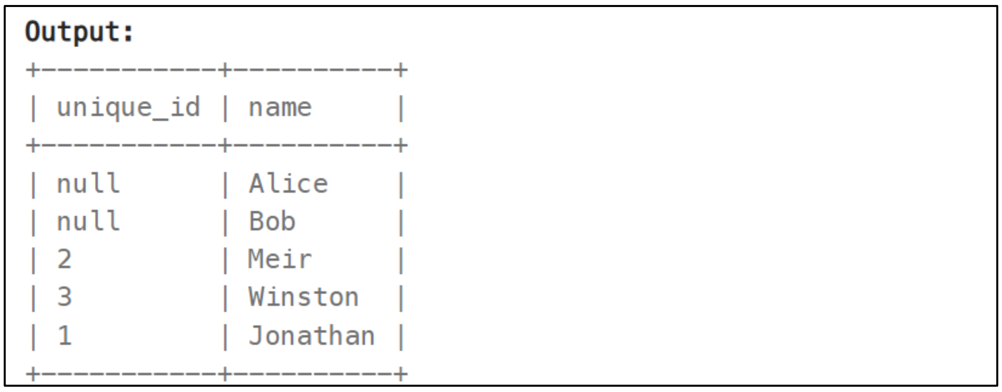
Example 1:

Input:

```
Employees table:
+----+----------+
| id | name     |
+----+----------+
| 1  | Alice    |
| 7  | Bob      |
| 11 | Meir     |
| 90 | Winston  |
| 3  | Jonathan |
+----+----------+
```

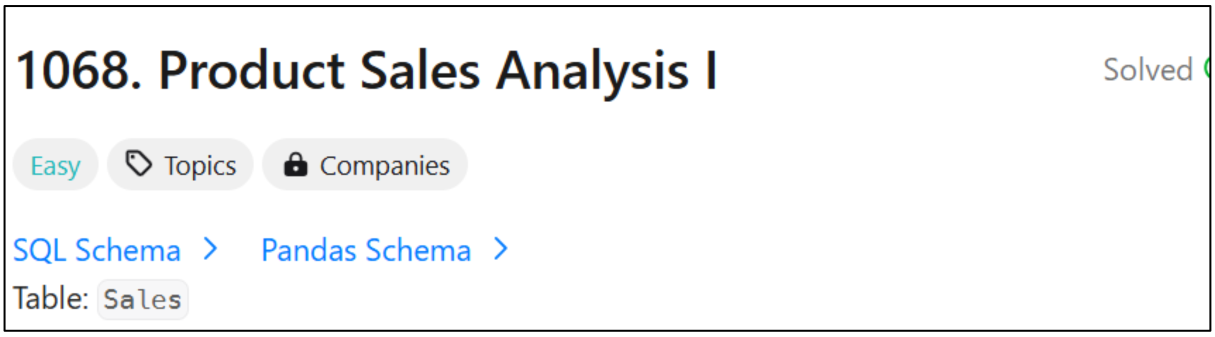


```
EmployeeUNI table:
+----+-----------+
| id | unique_id |
+----+-----------+
| 3  | 1         |
| 11 | 2         |
| 90 | 3         |
```



Output:

```
+-----------+----------+
| unique_id | name     |
+-----------+----------+
| null      | Alice    |
| null      | Bob      |
| 2         | Meir     |
| 3         | Winston  |
| 1         | Jonathan |
+-----------+----------+
```



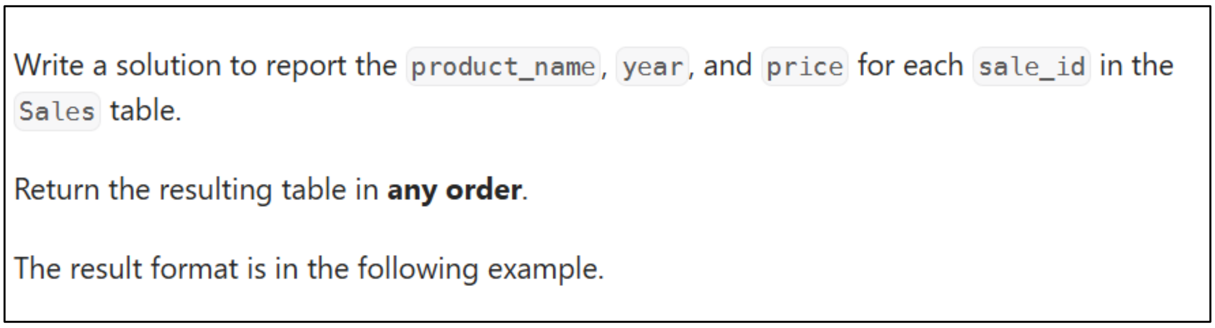
1068. Product Sales Analysis I

Solved

Easy Topics Companies

SQL Schema > Pandas Schema >

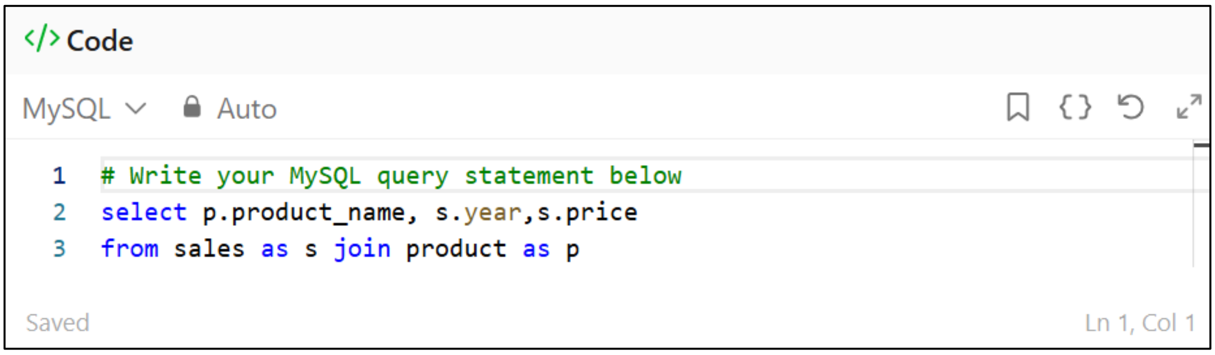
Table: `Sales`



Write a solution to report the `product_name`, `year`, and `price` for each `sale_id` in the `Sales` table.

Return the resulting table in **any order**.

The result format is in the following example.

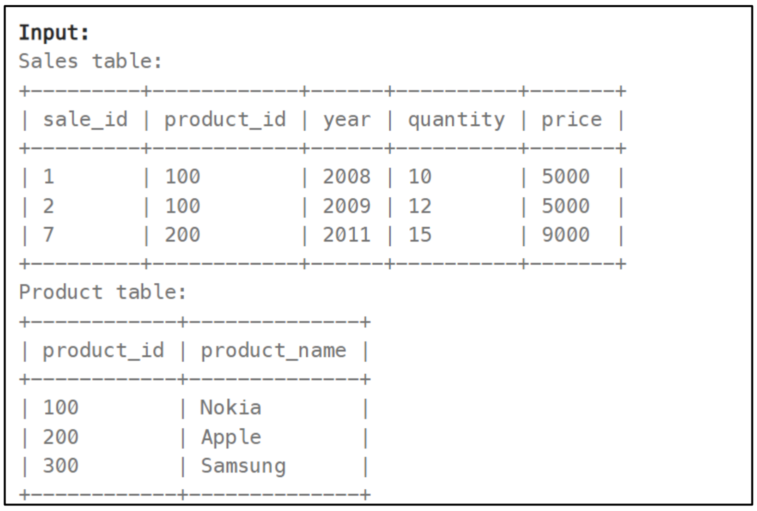


Code

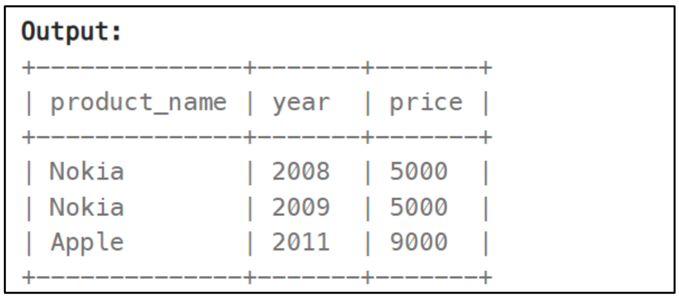
MySQL Auto

```sql
# Write your MySQL query statement below
select p.product_name, s.year,s.price
from sales as s join product as p
```

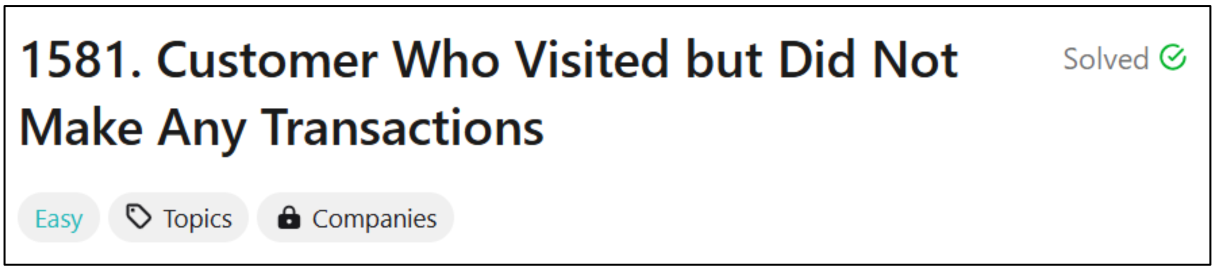
Saved Ln 1, Col 1



```
Input:
Sales table:
+---------+------------+------+----------+-------+
| sale_id | product_id | year | quantity | price |
+---------+------------+------+----------+-------+
| 1       | 100        | 2008 | 10       | 5000  |
| 2       | 100        | 2009 | 12       | 5000  |
| 7       | 200        | 2011 | 15       | 9000  |
+---------+------------+------+----------+-------+
Product table:
+------------+--------------+
| product_id | product_name |
+------------+--------------+
| 100        | Nokia        |
| 200        | Apple        |
| 300        | Samsung      |
+------------+--------------+
```



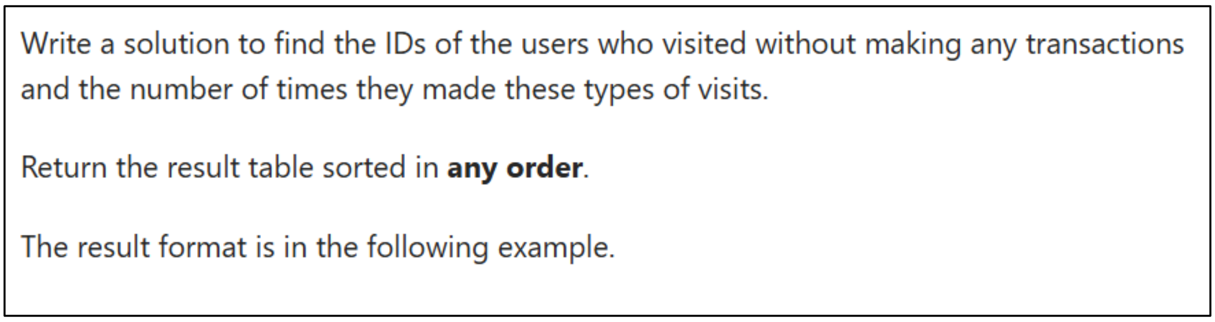
```
Output:
+--------------+-------+-------+
| product_name | year  | price |
+--------------+-------+-------+
| Nokia        | 2008  | 5000  |
| Nokia        | 2009  | 5000  |
| Apple        | 2011  | 9000  |
+--------------+-------+-------+
```



1581. Customer Who Visited but Did Not Make Any Transactions

Solved ✓

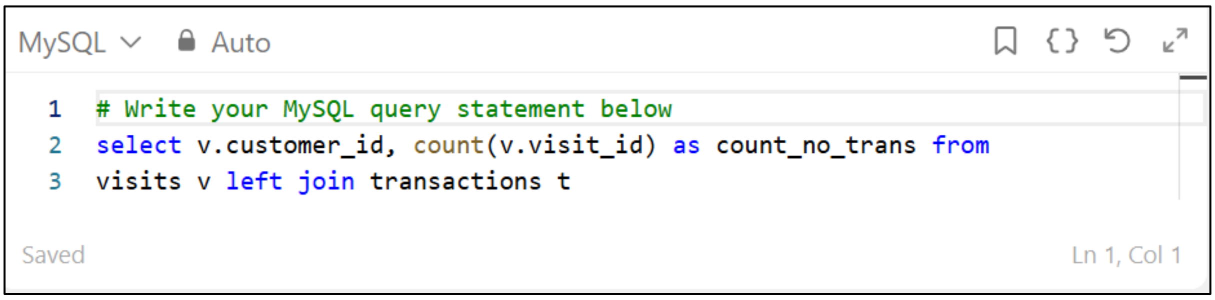
Easy · Topics · Companies



Write a solution to find the IDs of the users who visited without making any transactions and the number of times they made these types of visits.

Return the result table sorted in **any order**.

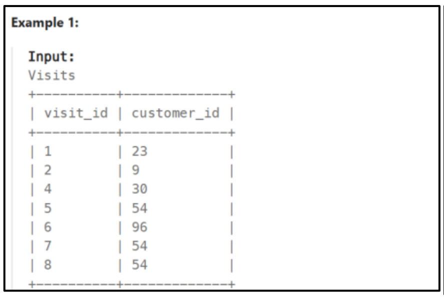
The result format is in the following example.



MySQL · Auto

```
# Write your MySQL query statement below
select v.customer_id, count(v.visit_id) as count_no_trans from
visits v left join transactions t
```

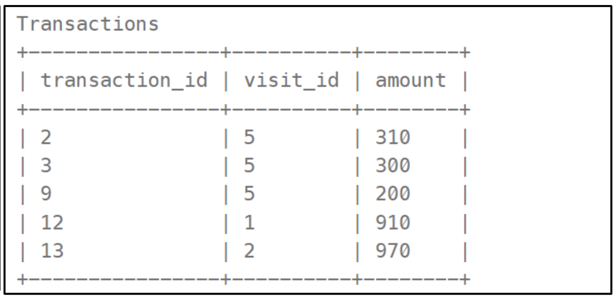
Saved · Ln 1, Col 1



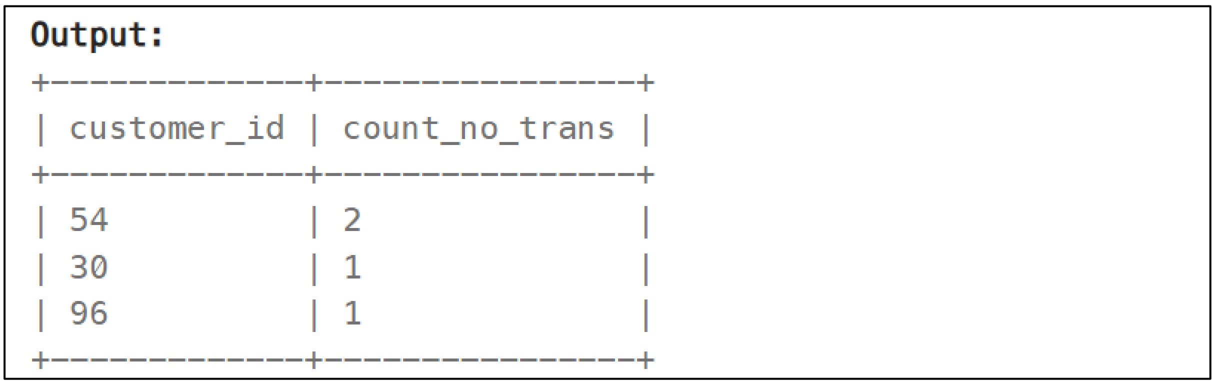
Example 1:

Input:

```
Visits
+----------+-------------+
| visit_id | customer_id |
+----------+-------------+
| 1        | 23          |
| 2        | 9           |
| 4        | 30          |
| 5        | 54          |
| 6        | 96          |
| 7        | 54          |
| 8        | 54          |
+----------+-------------+
```

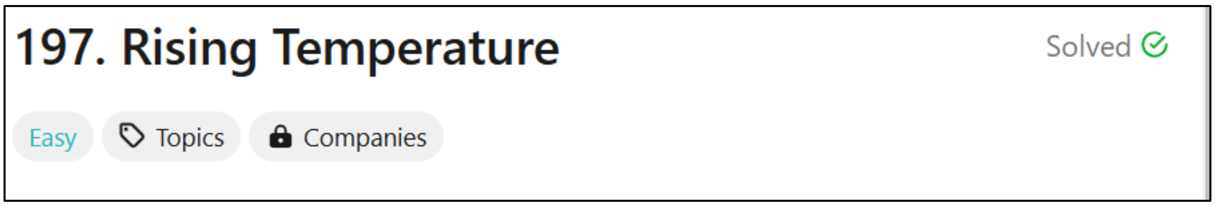


```
Transactions
+----------------+----------+--------+
| transaction_id | visit_id | amount |
+----------------+----------+--------+
| 2              | 5        | 310    |
| 3              | 5        | 300    |
| 9              | 5        | 200    |
| 12             | 1        | 910    |
| 13             | 2        | 970    |
+----------------+----------+--------+
```



Output:

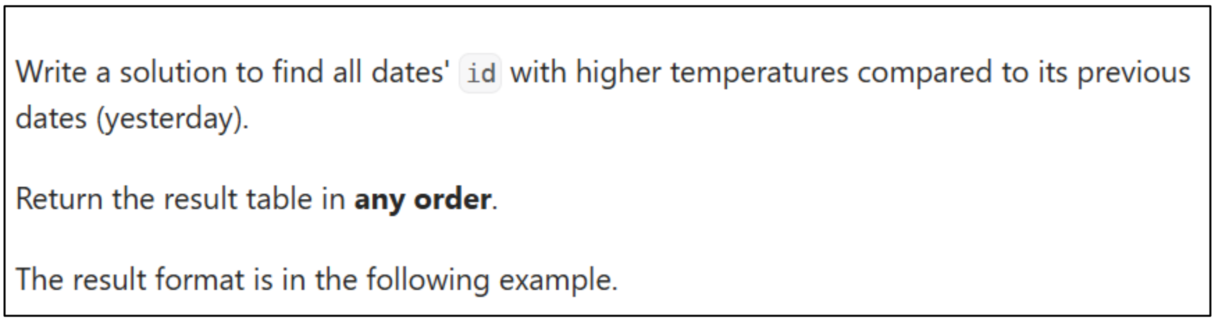
```
+-------------+----------------+
| customer_id | count_no_trans |
+-------------+----------------+
| 54          | 2              |
| 30          | 1              |
| 96          | 1              |
+-------------+----------------+
```



197. Rising Temperature

Solved ✓

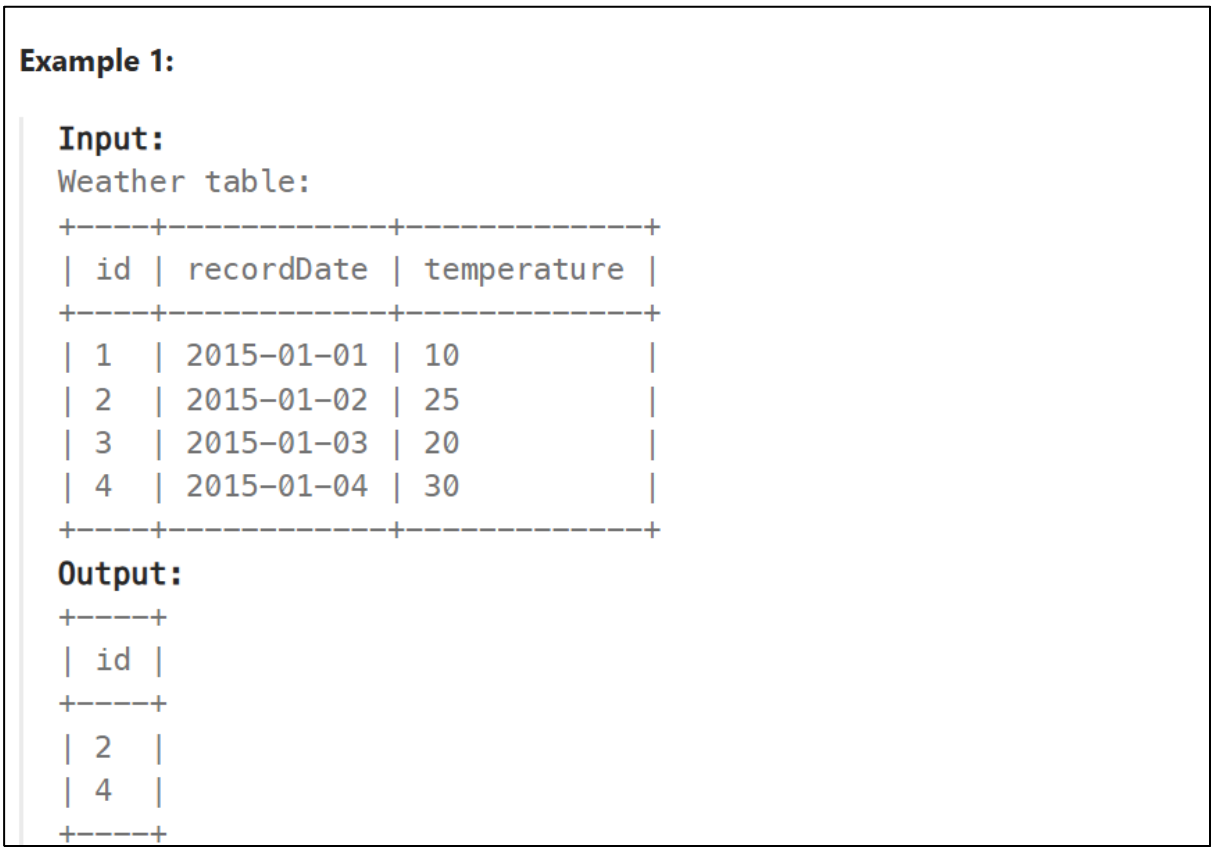
Easy · Topics · Companies



Write a solution to find all dates' `id` with higher temperatures compared to its previous dates (yesterday).

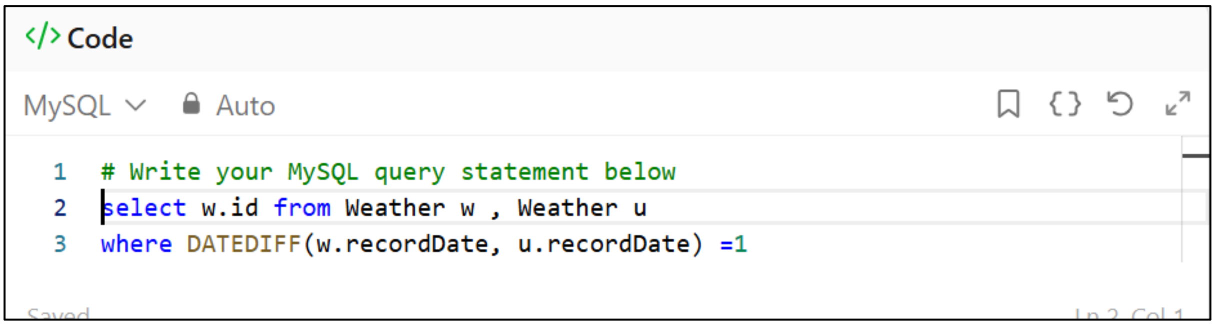
Return the result table in **any order**.

The result format is in the following example.



Example 1:

```
Input:
Weather table:
+----+------------+-------------+
| id | recordDate | temperature |
+----+------------+-------------+
| 1  | 2015-01-01 | 10          |
| 2  | 2015-01-02 | 25          |
| 3  | 2015-01-03 | 20          |
| 4  | 2015-01-04 | 30          |
+----+------------+-------------+
Output:
+----+
| id |
+----+
| 2  |
| 4  |
+----+
```



</> Code

MySQL · Auto

```sql
# Write your MySQL query statement below
select w.id from Weather w , Weather u
where DATEDIFF(w.recordDate, u.recordDate) =1
```

Saved · Ln 2, Col 1