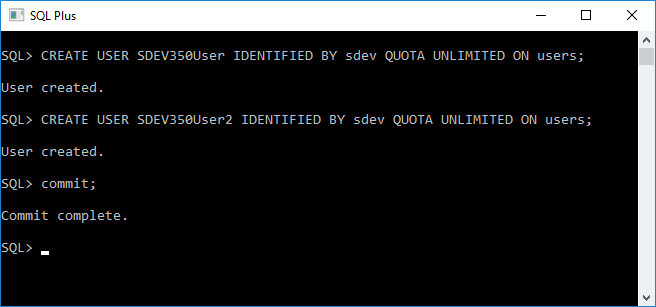
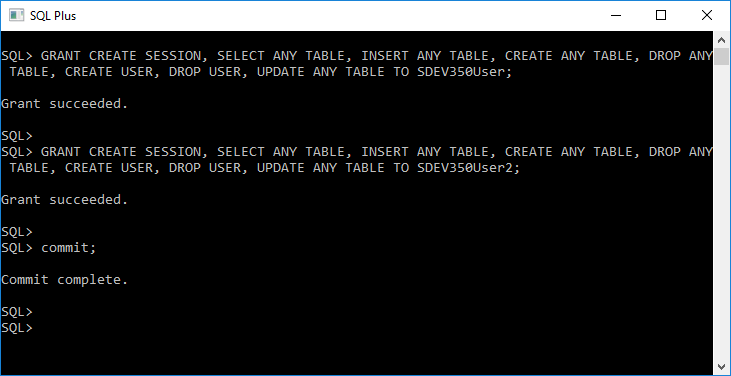
I started off by creating two users, SDEV350User and SDEV350User2, both of which have unlimited quota on the users tablespace. These users are being made using the SYS account.



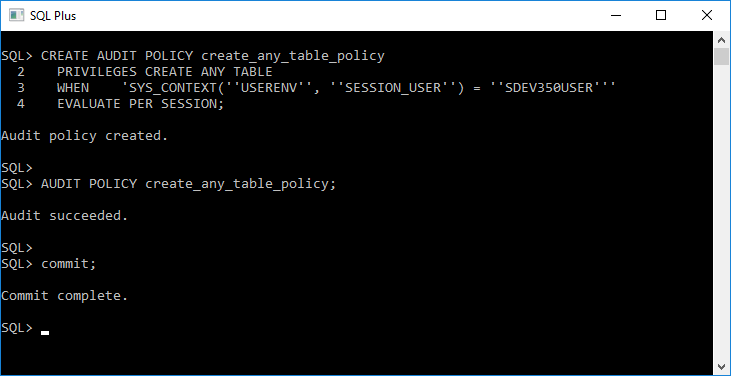
Then I grant the permissions necessary to create sessions, create any table, drop any table, create users, drop users, and update any table. These grants are being made using the SYS account.



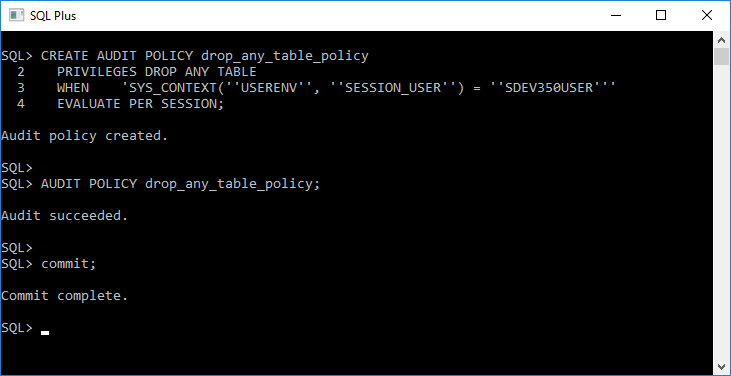
Audit policies are a great way of being able to enable accountability for a user's actions. Knowing this, user’s may be hesitant to try and do something that they know is being tracked by an upper level user. Audit policies are also a great way to monitor and gather data about specific database activities that may be going on that you should be aware about. Database administrators are able to check numerous statistics such as which tables are being updated, different operations being performed, or even how many concurrent users connect at peak times.

We then begin to create our audit policy to keep track of user actions. We first implement a CREATE ANY TABLE policy which we will use to audit that user to see if any tables are being made under that user's schema. This is a great way of monitoring who is making tables and for what purpose in the database.

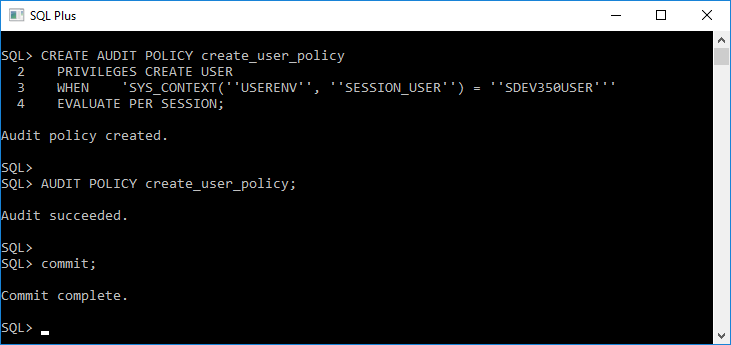
All of the following policies are being made using the SYS account.



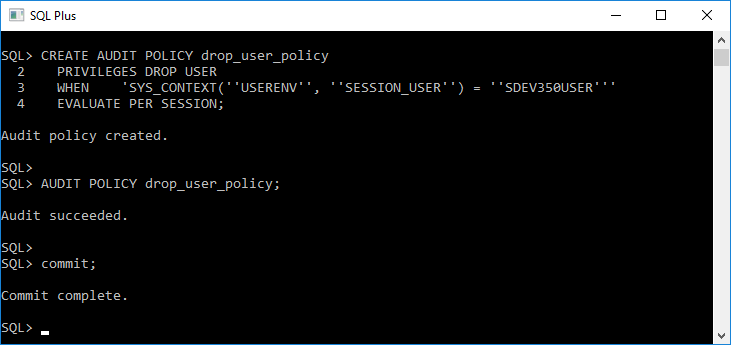
I then created the second audit policy, which is the DROP ANY TABLE policy. This is very important in database management simply because if a table happens to disappear from your database, you know exactly who to blame as per the audit and which tables are affected.



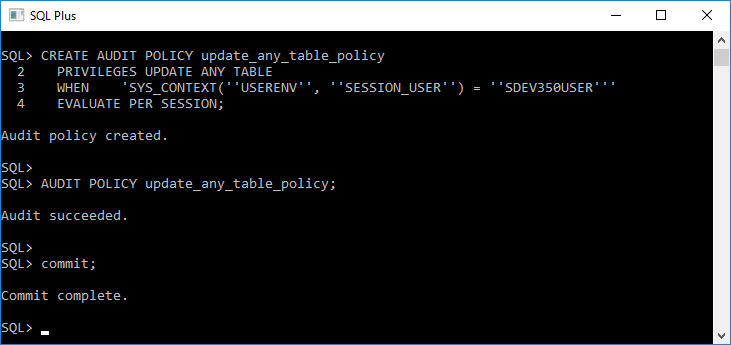
The third audit policy is CREATE USER. What we track here is when our user, SDEV350USER, creates another user under his credentials, it will track and let us see who the user is and the time and date of the users creation. Date and time is not specific to this policy, all policies do have a date and time when auditing. We will also be able to use sql commands to see the privileges of that user and take necessary action against them if needed.



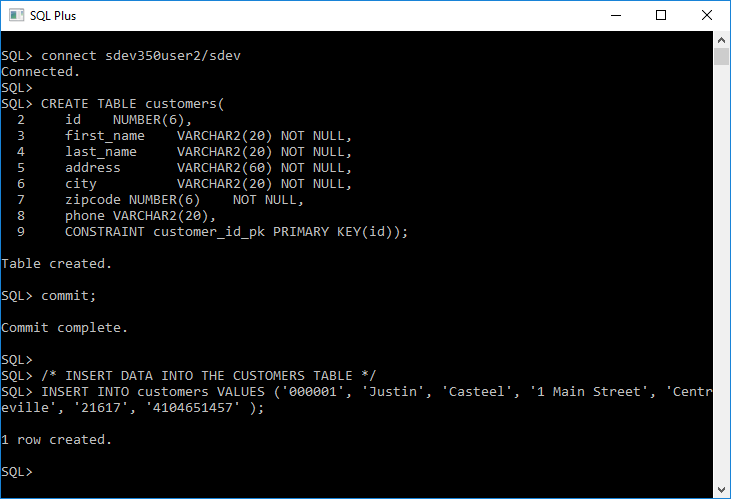
My fourth audit policy is DROP USER. What this does is it audits the use of drop user, so when another user drops one of his fellow users from the database, this keeps track of all information needed. This could be potentially dangerous if privileges were to be used to be able to drop an admin account or any other user who has higher permissions than yourself. If an admin account is dropped by a lower user than it may be more difficult to regain access to the system and find out who the culprit is and what other potential threats that user may have. If the user is able to drop an admin account then it wouldn’t be unlikely that they would be able to access and clear the audit trail history so they couldn’t be traced once the admin regained access.



The last audit policy, UPDATE ANY TABLE, is used to track the specific user that may be updating the information in a table. This also includes information from another user’s tables as well as their own. This is a great example of how an audit policy could protect the integrity of a database. For example, if you ran your own business and you wanted to keep track of your employees information along with their hourly wages. You wouldn’t want a user to go in and update the information in that table without you immediately knowing about it, right? This is perfect for those quick audits where you want to see if something has been updated in a table and maybe it shouldn’t have been.



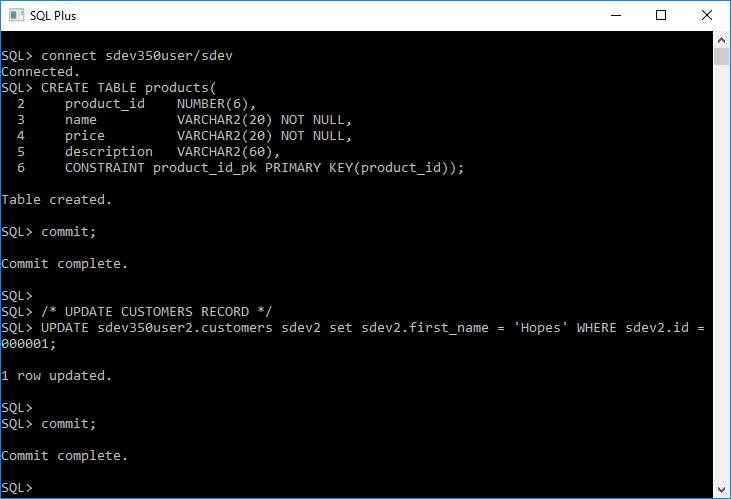
We then connect to the sdev350user2. We do this because we want to create a table on this users schema and add information so that when we update that information from our other user, sdev350user, it will show up as an UPDATE audit. We also do this so that when we use the drop table command it will audit this as well from the sdev350user user.



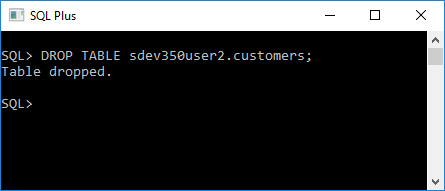
We then connect as the sdev350user and create a table on that user’s schema that will show on the audit. We then update the previously made information in the table “customers” created by sdev350user2. This was the trickiest part of this entire assignment and it turned out to be that I didn’t grant all of the permissions required to update a row in another user’s table. The permissions were:

SELECT ANY TABLE and INSERT ANY TABLE

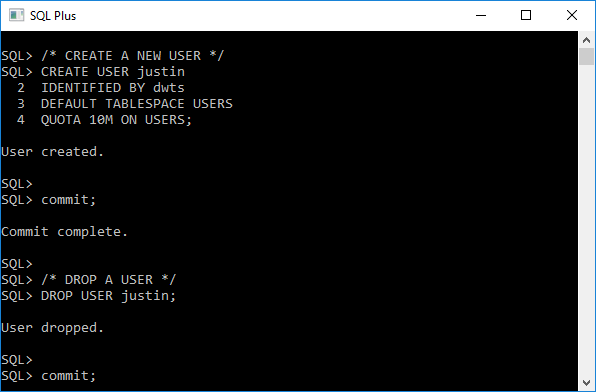
Once I assigned these to the user I had no problem at all with updating table information.



For auditing purposes I then drop the sdev350user2 customers table.

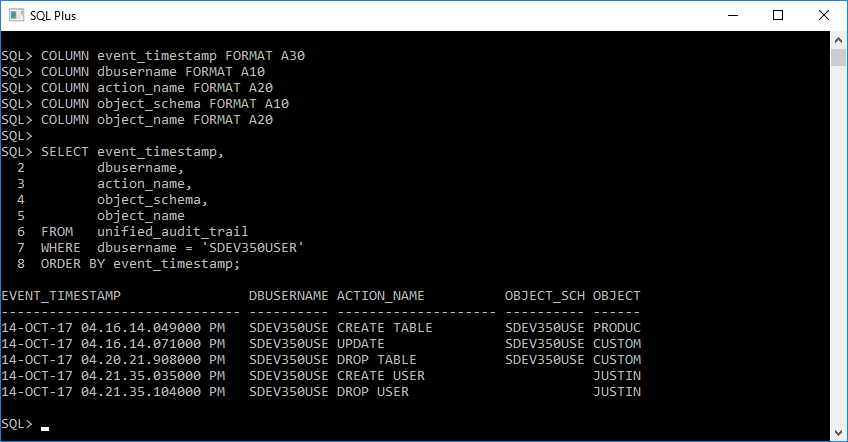


For the final audit to show up I needed to create a new user under the sdev350user and then drop that same user. So in this case we create a user with the name of justin and then drop that user afterwards. This would show as CREATE USER and DROP USER audit when we query our unified\_audit\_trail and check all of our audits we have done.



I then format the columns so that it looks readable and we query the unified\_audit\_trail to show the audits that we have just done above. This is an amazing tool and shows the date and time of the action and the user who performed those actions.

Note: You must be using the sys account for the following query to execute.



Auditing is an amazing way to keep the integrity of your database in tact and to remove any of those suspicious users who may be performing actions they shouldn’t be. It keeps the company operating in a fair and honest manner so and also keeps users aware that they are being tracked and they can be held accountable for their actions. Audits are a great way to keep track of your financial statements and information. With some companies there may be specific laws that you would need to follow such as environmental laws or checking to see if a company is using government funds properly and legally. Auditing is a great way to stay compliant with regulations and to keep track of what is happening in the company you’re working for.

Below is the successful running of my script.sql file which contains all of the above SQL statements.

