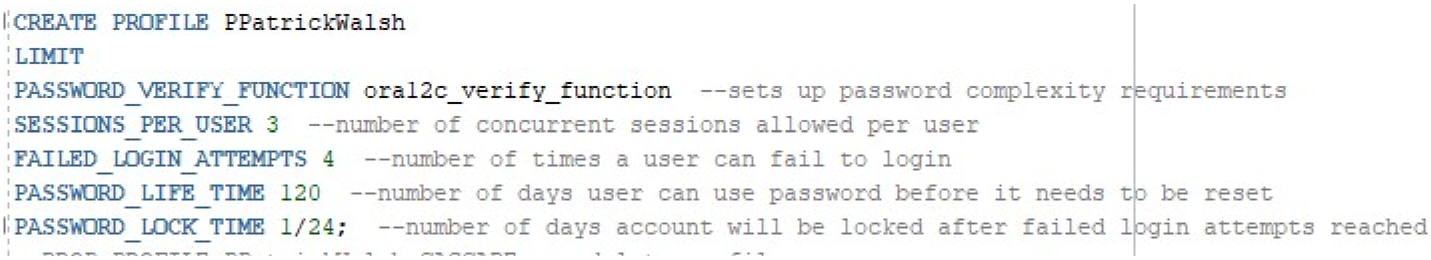
SDEV 350 Lab 3

Student: Patrick Walsh
Date: 4/9/2021
Course: SDEV 350 7380
Professor: Reginald Haseltine

1. Create a unique Profile based on the following requirements:
   a) Password complexity should meet requirements for Ora12 Verify function.
   b) User may have up to 3 concurrent sessions.
   c) User may have up to 4 consecutive failed attempts to log in before the account is locked.
   d) User may wait up till 120 days before their password must be changed.
   e) User account will be locked for 1 hours after the specified number of consecutive failed login attempts.
   f) Default values for other Profile parameters is acceptable.
   g) You should name the Profile PFirstnameLastname where Lastname and Firstname are your First and Lastname.

Fig 1a

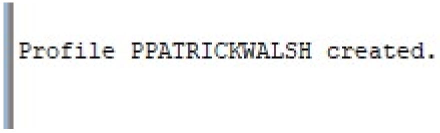
Code:



```
CREATE PROFILE PPatrickWalsh
LIMIT
PASSWORD_VERIFY_FUNCTION ora12c_verify_function  --sets up password complexity requirements
SESSIONS_PER_USER 3  --number of concurrent sessions allowed per user
FAILED_LOGIN_ATTEMPTS 4  --number of times a user can fail to login
PASSWORD_LIFE_TIME 120  --number of days user can use password before it needs to be reset
PASSWORD_LOCK_TIME 1/24;  --number of days account will be locked after failed login attempts reached
```

Fig 1b

Output:

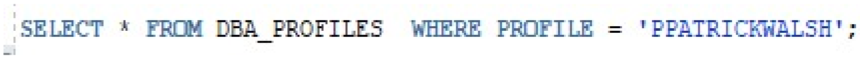


```
Profile PPATRICKWALSH created.
```

2. Verify your Profile was successfully created by Creating and executing a SQL statement querying the appropriate Data Dictionaryobjects.

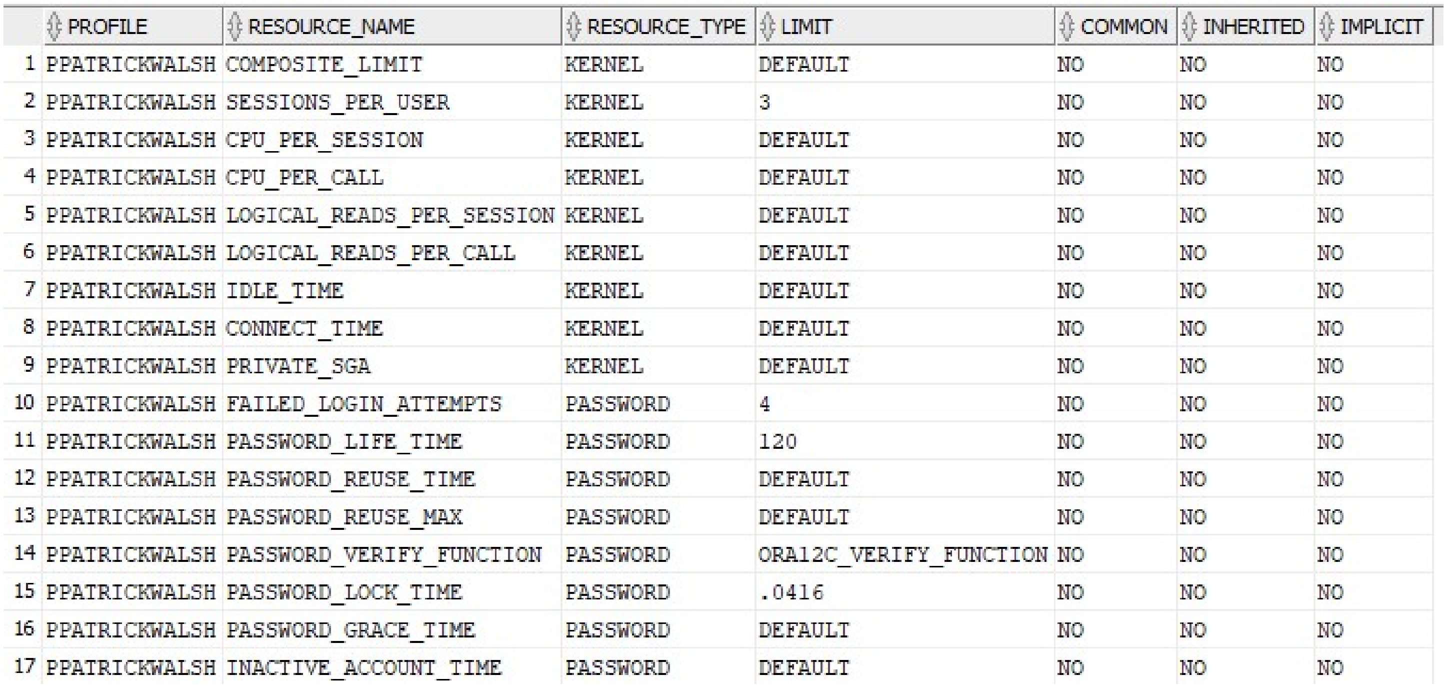
Fig 2a

Code:



```
SELECT * FROM DBA_PROFILES  WHERE PROFILE = 'PPATRICKWALSH';
```

Fig 2b
Output:

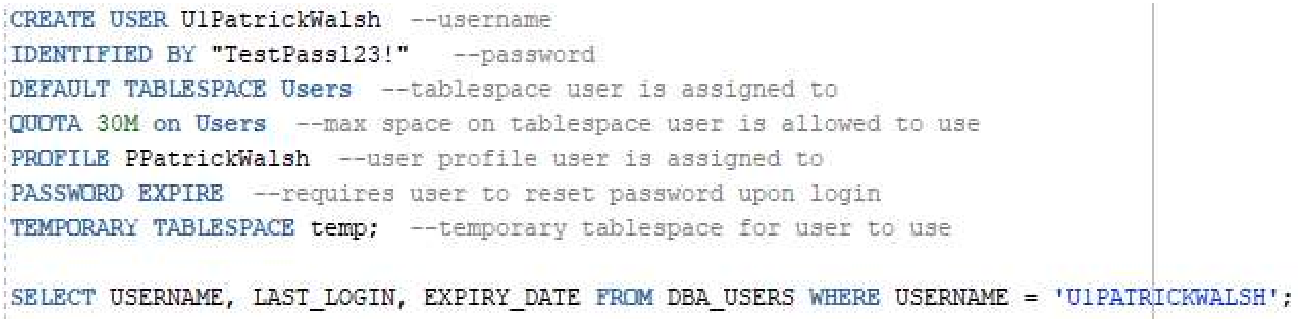


	PROFILE	RESOURCE_NAME	RESOURCE_TYPE	LIMIT	COMMON	INHERITED	IMPLICIT
1	PPATRICKWALSH	COMPOSITE_LIMIT	KERNEL	DEFAULT	NO	NO	NO
2	PPATRICKWALSH	SESSIONS_PER_USER	KERNEL	3	NO	NO	NO
3	PPATRICKWALSH	CPU_PER_SESSION	KERNEL	DEFAULT	NO	NO	NO
4	PPATRICKWALSH	CPU_PER_CALL	KERNEL	DEFAULT	NO	NO	NO
5	PPATRICKWALSH	LOGICAL_READS_PER_SESSION	KERNEL	DEFAULT	NO	NO	NO
6	PPATRICKWALSH	LOGICAL_READS_PER_CALL	KERNEL	DEFAULT	NO	NO	NO
7	PPATRICKWALSH	IDLE_TIME	KERNEL	DEFAULT	NO	NO	NO
8	PPATRICKWALSH	CONNECT_TIME	KERNEL	DEFAULT	NO	NO	NO
9	PPATRICKWALSH	PRIVATE_SGA	KERNEL	DEFAULT	NO	NO	NO
10	PPATRICKWALSH	FAILED_LOGIN_ATTEMPTS	PASSWORD	4	NO	NO	NO
11	PPATRICKWALSH	PASSWORD_LIFE_TIME	PASSWORD	120	NO	NO	NO
12	PPATRICKWALSH	PASSWORD_REUSE_TIME	PASSWORD	DEFAULT	NO	NO	NO
13	PPATRICKWALSH	PASSWORD_REUSE_MAX	PASSWORD	DEFAULT	NO	NO	NO
14	PPATRICKWALSH	PASSWORD_VERIFY_FUNCTION	PASSWORD	ORA12C_VERIFY_FUNCTION	NO	NO	NO
15	PPATRICKWALSH	PASSWORD_LOCK_TIME	PASSWORD	.0416	NO	NO	NO
16	PPATRICKWALSH	PASSWORD_GRACE_TIME	PASSWORD	DEFAULT	NO	NO	NO
17	PPATRICKWALSH	INACTIVE_ACCOUNT_TIME	PASSWORD	DEFAULT	NO	NO	NO

3. Create 2 users assign them to the Permanent Tablespace of Users with a Quota of 30M. Assign the new users the Profile you established in Step 1 of this lab. Be sure to expire their passwords upon creation.
Name the users as follows:

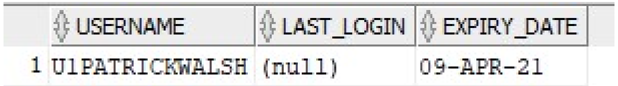
a) U1FirstnameLastname
b) U2FirstnameLastname Where Firstname and Lastname are your first and lastname.

Fig 3a
Code(1):



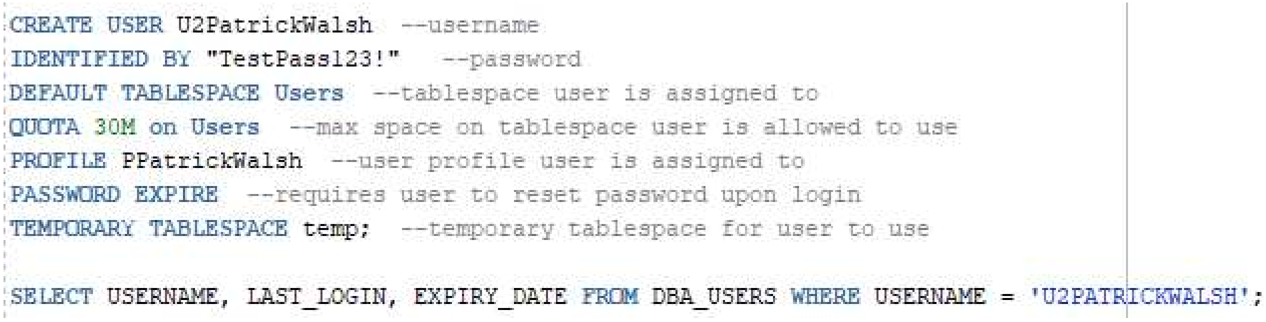
```
CREATE USER U1PatrickWalsh  --username
IDENTIFIED BY "TestPass123!"   --password
DEFAULT TABLESPACE Users  --tablespace user is assigned to
QUOTA 30M on Users  --max space on tablespace user is allowed to use
PROFILE PPatrickWalsh  --user profile user is assigned to
PASSWORD EXPIRE  --requires user to reset password upon login
TEMPORARY TABLESPACE temp;  --temporary tablespace for user to use

SELECT USERNAME, LAST_LOGIN, EXPIRY_DATE FROM DBA_USERS WHERE USERNAME = 'U1PATRICKWALSH';
```

Fig 3b
Output(1):



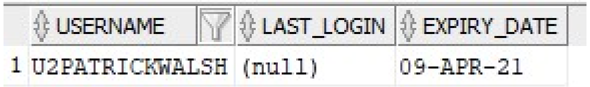
	USERNAME	LAST_LOGIN	EXPIRY_DATE
1	U1PATRICKWALSH	(null)	09-APR-21

Fig 3c
Code(2):



```
CREATE USER U2PatrickWalsh  --username
IDENTIFIED BY "TestPass123!"   --password
DEFAULT TABLESPACE Users  --tablespace user is assigned to
QUOTA 30M on Users  --max space on tablespace user is allowed to use
PROFILE PPatrickWalsh  --user profile user is assigned to
PASSWORD EXPIRE  --requires user to reset password upon login
TEMPORARY TABLESPACE temp;  --temporary tablespace for user to use

SELECT USERNAME, LAST_LOGIN, EXPIRY_DATE FROM DBA_USERS WHERE USERNAME = 'U2PATRICKWALSH';
```

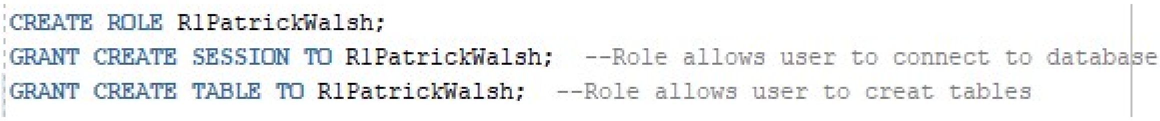
Fig 3d
Output(2):



```
  USERNAME         LAST_LOGIN  EXPIRY_DATE
1 U2PATRICKWALSH   (null)      09-APR-21
```

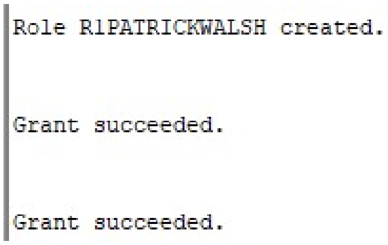
4. Create a role allowing users assigned to be able to connect to the database and create tables. Name this R1FirstnameLastname where Firstname and Lastname are your first and lastname.

Fig 4a
Code:



```
CREATE ROLE R1PatrickWalsh;
GRANT CREATE SESSION TO R1PatrickWalsh;  --Role allows user to connect to database
GRANT CREATE TABLE TO R1PatrickWalsh;  --Role allows user to creat tables
```

Fig 4b
Output:

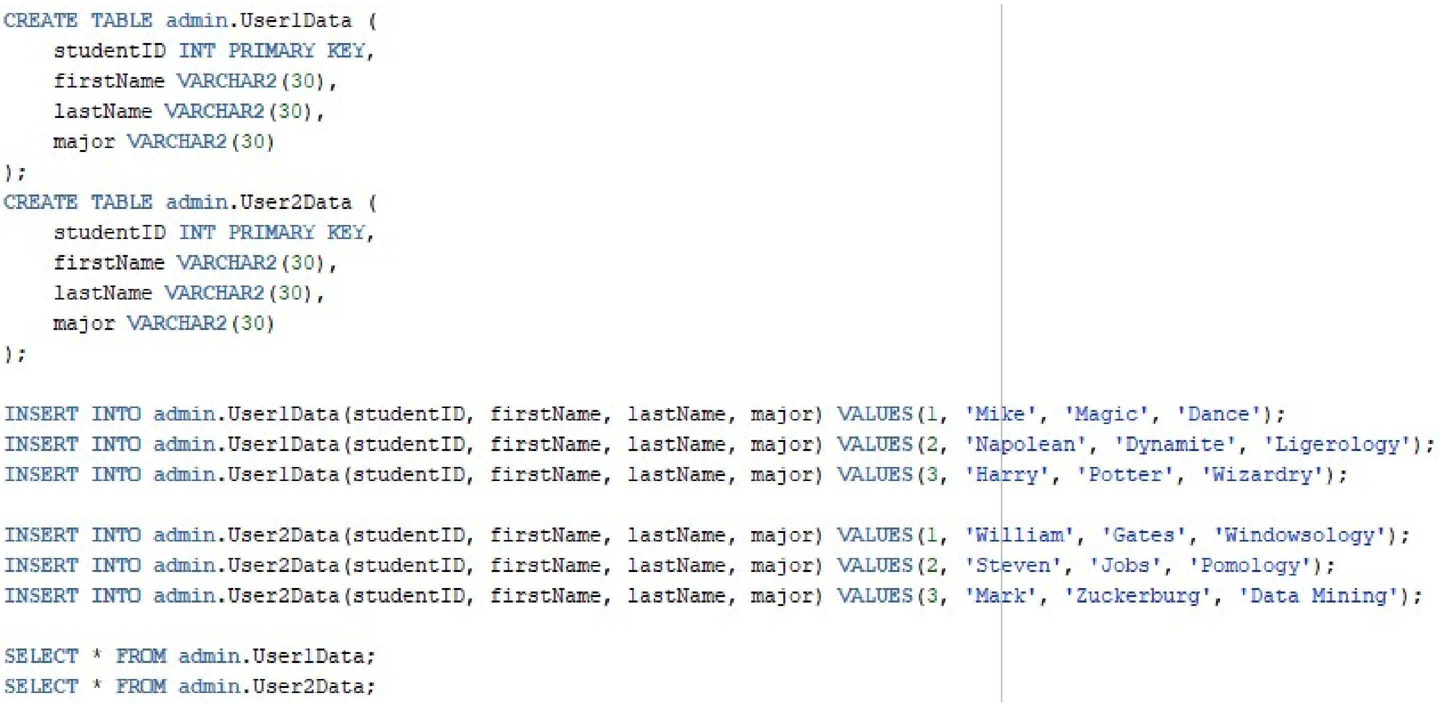


```
Role R1PATRICKWALSH created.


Grant succeeded.


Grant succeeded.
```

5. Create two tables in your (root/admin) schema. Name them User1Data and User2Data. The tables should contain a primary key and 3 additional columns of your choice. Insert 3 records into each table.

Fig 5a

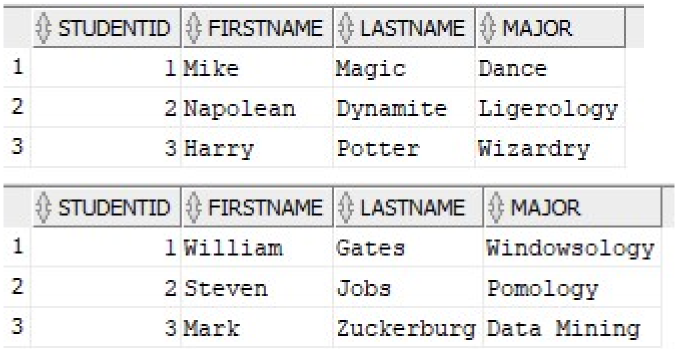
Code:



```
CREATE TABLE admin.User1Data (
    studentID INT PRIMARY KEY,
    firstName VARCHAR2(30),
    lastName VARCHAR2(30),
    major VARCHAR2(30)
);
CREATE TABLE admin.User2Data (
    studentID INT PRIMARY KEY,
    firstName VARCHAR2(30),
    lastName VARCHAR2(30),
    major VARCHAR2(30)
);

INSERT INTO admin.User1Data(studentID, firstName, lastName, major) VALUES(1, 'Mike', 'Magic', 'Dance');
INSERT INTO admin.User1Data(studentID, firstName, lastName, major) VALUES(2, 'Napolean', 'Dynamite', 'Ligerology');
INSERT INTO admin.User1Data(studentID, firstName, lastName, major) VALUES(3, 'Harry', 'Potter', 'Wizardry');

INSERT INTO admin.User2Data(studentID, firstName, lastName, major) VALUES(1, 'William', 'Gates', 'Windowsology');
INSERT INTO admin.User2Data(studentID, firstName, lastName, major) VALUES(2, 'Steven', 'Jobs', 'Pomology');
INSERT INTO admin.User2Data(studentID, firstName, lastName, major) VALUES(3, 'Mark', 'Zuckerburg', 'Data Mining');

SELECT * FROM admin.User1Data;
SELECT * FROM admin.User2Data;
```

Fig 5b

Output:



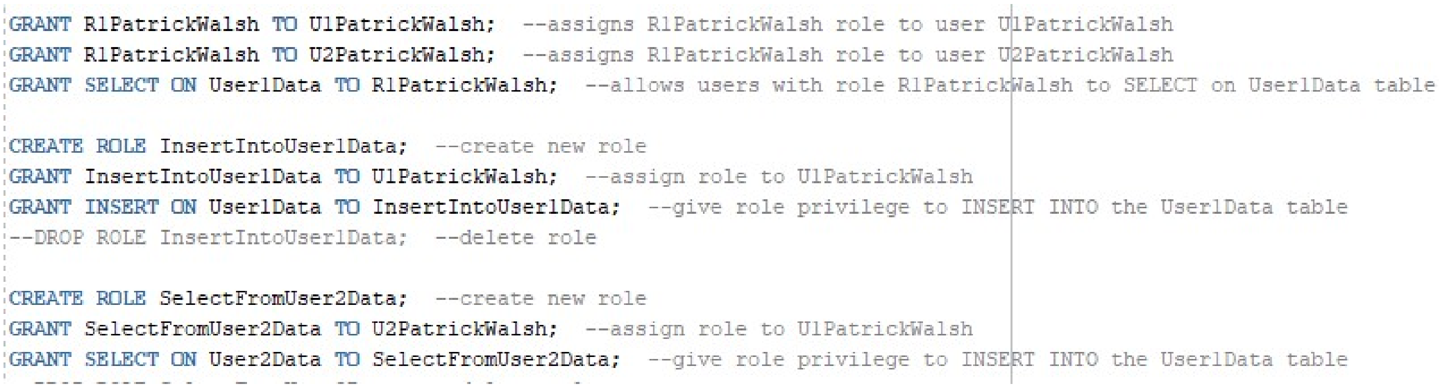
	STUDENTID	FIRSTNAME	LASTNAME	MAJOR
1	1	Mike	Magic	Dance
2	2	Napolean	Dynamite	Ligerology
3	3	Harry	Potter	Wizardry

	STUDENTID	FIRSTNAME	LASTNAME	MAJOR
1	1	William	Gates	Windowsology
2	2	Steven	Jobs	Pomology
3	3	Mark	Zuckerburg	Data Mining

6. Provide privileges for U1FirstnameLastname and U2FirstnameLastname to be able to connect to the database and create tables. Be sure to use security best practices when assigning these privileges. In addition, provide one user the privileges to select from User1Data and Insert into User1Data. Provide the other user the privileges to select from User1Data and User2Data.

Fig 6a

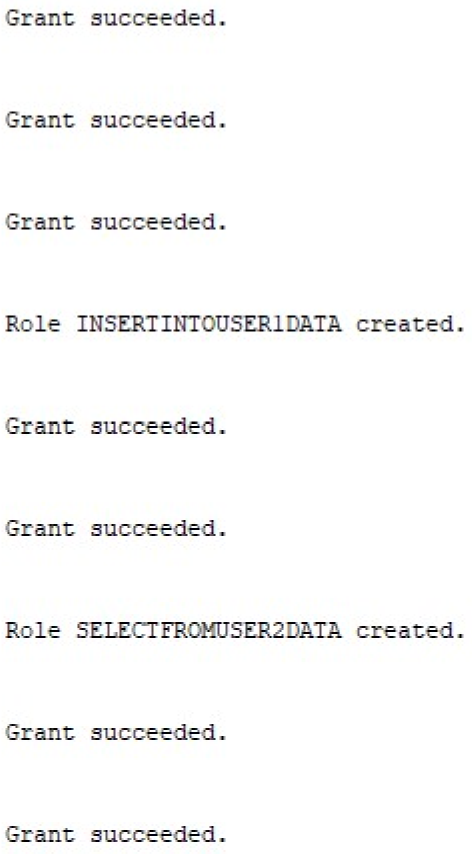
Code:



```
GRANT R1PatrickWalsh TO U1PatrickWalsh;  --assigns R1PatrickWalsh role to user U1PatrickWalsh
GRANT R1PatrickWalsh TO U2PatrickWalsh;  --assigns R1PatrickWalsh role to user U2PatrickWalsh
GRANT SELECT ON User1Data TO R1PatrickWalsh;  --allows users with role R1PatrickWalsh to SELECT on User1Data table

CREATE ROLE InsertIntoUser1Data;  --create new role
GRANT InsertIntoUser1Data TO U1PatrickWalsh;  --assign role to U1PatrickWalsh
GRANT INSERT ON User1Data TO InsertIntoUser1Data;  --give role privilege to INSERT INTO the User1Data table
--DROP ROLE InsertIntoUser1Data;  --delete role

CREATE ROLE SelectFromUser2Data;  --create new role
GRANT SelectFromUser2Data TO U2PatrickWalsh;  --assign role to U1PatrickWalsh
GRANT SELECT ON User2Data TO SelectFromUser2Data;  --give role privilege to INSERT INTO the User1Data table
```

Fig 6b

Output:

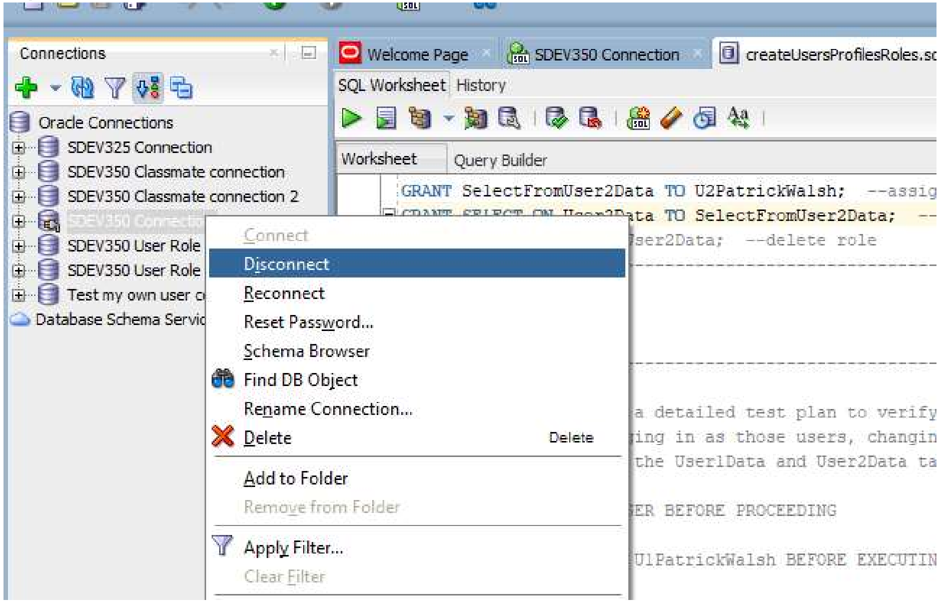


```
Grant succeeded.


Grant succeeded.


Grant succeeded.


Role INSERTINTOUSER1DATA created.


Grant succeeded.


Grant succeeded.


Role SELECTFROMUSER2DATA created.


Grant succeeded.


Grant succeeded.
```

7. Prepare and execute a detailed test plan to verify the two users have all the privileges they need but no additional privileges. Be sure to test by logging in as those users, changing their passwords, connecting, creating table and then using the assigned privileges in the User1Data and User2Data tables and performing and documenting other tests as required.

LOGOUT FROM ADMIN USER BEFORE PROCEEDING:

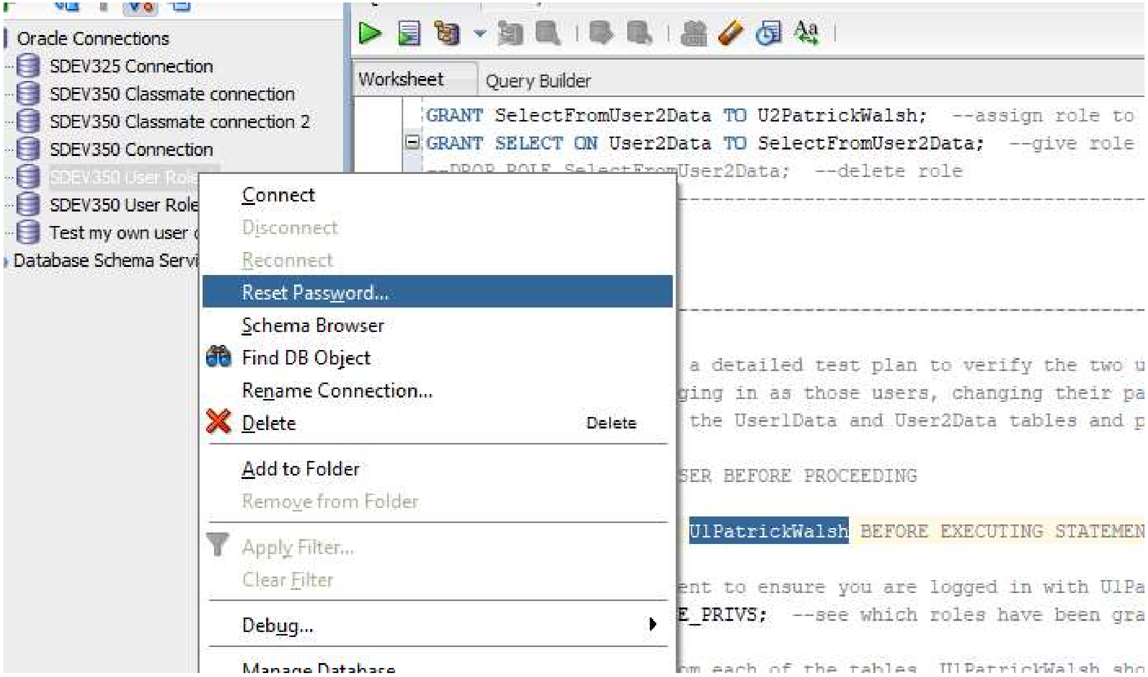
Fig 7a



```
GRANT SelectFromUser2Data TO U2PatrickWalsh;  --assig
GRANT SELECT ON User2Data TO SelectFromUser2Data;  --
ser2Data;  --delete role
---------------------------------
---------------------------------
a detailed test plan to verify
ing in as those users, changin
the User1Data and User2Data ta
ER BEFORE PROCEEDING
U1PatrickWalsh BEFORE EXECUTIN
```

Then reconnect to database as U1PatrickWalsh. Reset password upon first login:

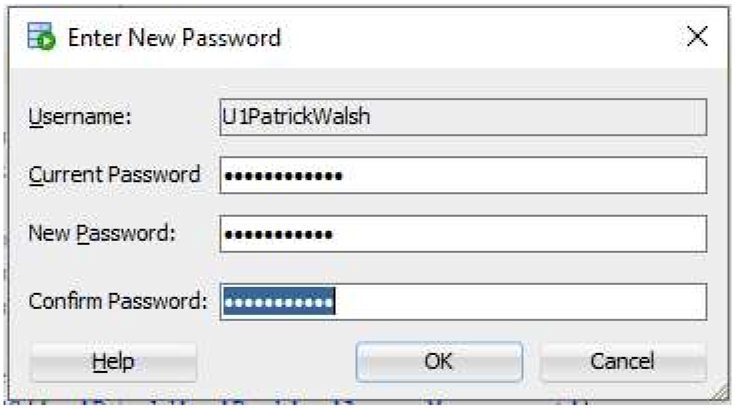
Fig 7b



```
GRANT SelectFromUser2Data TO U2PatrickWalsh;  --assign role to
GRANT SELECT ON User2Data TO SelectFromUser2Data;  --give role
--DROP ROLE SelectFromUser2Data;  --delete role
-------------------------------------------
-------------------------------------------
a detailed test plan to verify the two u
ging in as those users, changing their pa
the User1Data and User2Data tables and p
SER BEFORE PROCEEDING
U1PatrickWalsh BEFORE EXECUTING STATEMEN
ent to ensure you are logged in with U1Pa
E_PRIVS;  --see which roles have been gra
om each of the tables. U1PatrickWalsh sho
```

Enter current password = ‘TestPass123!’
New password = ‘NewPass123!’
Confirm new password = ‘NewPass123!’

Fig 7c



Enter New Password

Username: U1PatrickWalsh

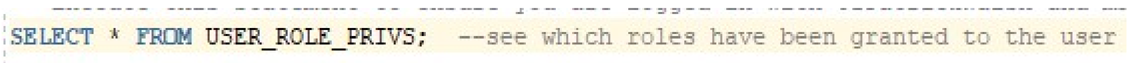
Current Password •••••••••••••

New Password: ••••••••••••

Confirm Password: ••••••••••••

Check user is logged in as U1PatrickWalsh and that you see the appropriate privileges:

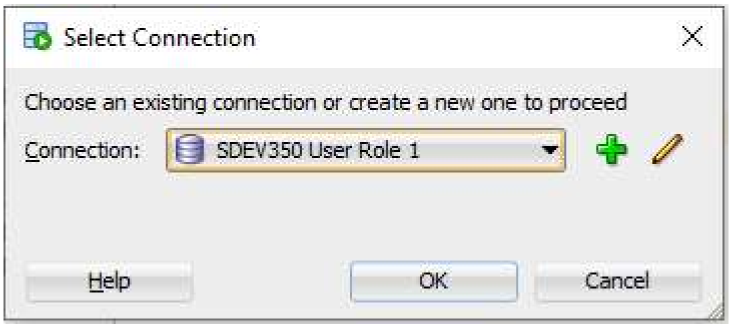
Fig 7d
Code:



```
SELECT * FROM USER_ROLE_PRIVS;  --see which roles have been granted to the user
```

Select connection that uses U1PatrickWalsh user:

Fig 7e

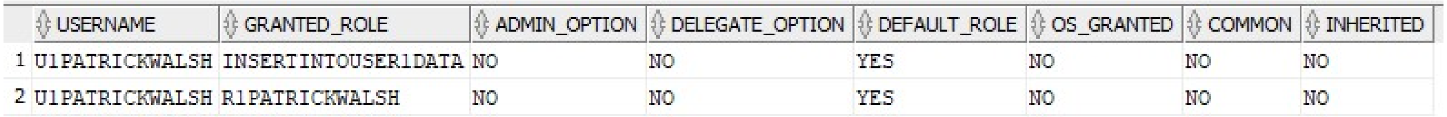


Select Connection

Choose an existing connection or create a new one to proceed

Connection: SDEV350 User Role 1

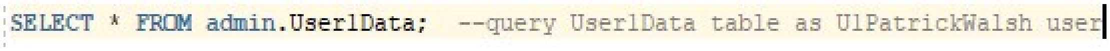
Fig 7f
Output:



	USERNAME	GRANTED_ROLE	ADMIN_OPTION	DELEGATE_OPTION	DEFAULT_ROLE	OS_GRANTED	COMMON	INHERITED
1	U1PATRICKWALSH	INSERTINTOUSER1DATA	NO	NO	YES	NO	NO	NO
2	U1PATRICKWALSH	R1PATRICKWALSH	NO	NO	YES	NO	NO	NO

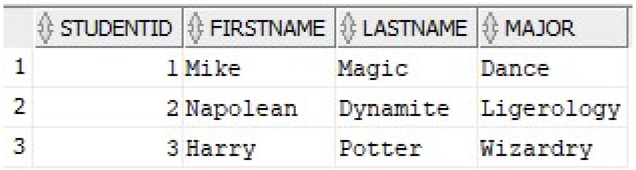
Now try to query from each of the tables. U1PatrickWalsh should be able to query User1Data but NOT User2Data:

Fig 7g
Code(1):



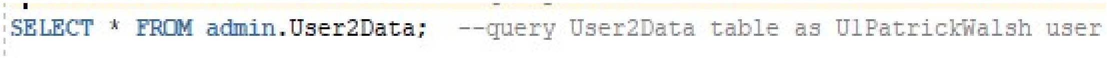
```
SELECT * FROM admin.User1Data;  --query User1Data table as U1PatrickWalsh user
```

Fig 7h
Output(1):



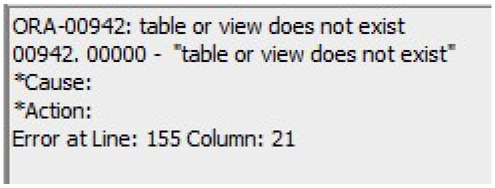
	STUDENTID	FIRSTNAME	LASTNAME	MAJOR
1	1	Mike	Magic	Dance
2	2	Napolean	Dynamite	Ligerology
3	3	Harry	Potter	Wizardry

Fig 7i
Code(2):



```
SELECT * FROM admin.User2Data;  --query User2Data table as U1PatrickWalsh user
```

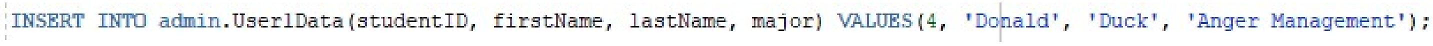
Fig 7j
Output(2):



```
ORA-00942: table or view does not exist
00942. 00000 -  "table or view does not exist"
*Cause:
*Action:
Error at Line: 155 Column: 21
```

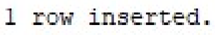
Now try to insert data into each table. U1PatrickWalsh should be able to insert into User1Data but NOT User2Data:

Fig 7k
Code(1):



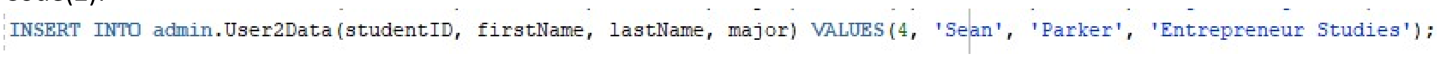
```
INSERT INTO admin.User1Data(studentID, firstName, lastName, major) VALUES(4, 'Donald', 'Duck', 'Anger Management');
```

Fig 7l
Output(1):



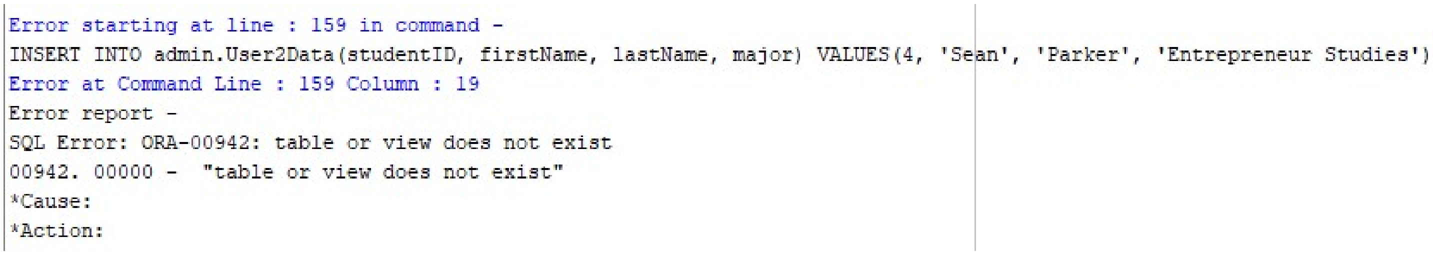
```
1 row inserted.
```

Fig 7m
Code(2):



```
INSERT INTO admin.User2Data(studentID, firstName, lastName, major) VALUES(4, 'Sean', 'Parker', 'Entrepreneur Studies');
```

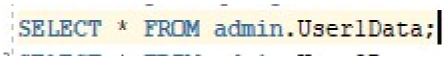
Fig 7n
Output(2):



```
Error starting at line : 159 in command -
INSERT INTO admin.User2Data(studentID, firstName, lastName, major) VALUES(4, 'Sean', 'Parker', 'Entrepreneur Studies')
Error at Command Line : 159 Column : 19
Error report -
SQL Error: ORA-00942: table or view does not exist
00942. 00000 -  "table or view does not exist"
*Cause:
*Action:
```

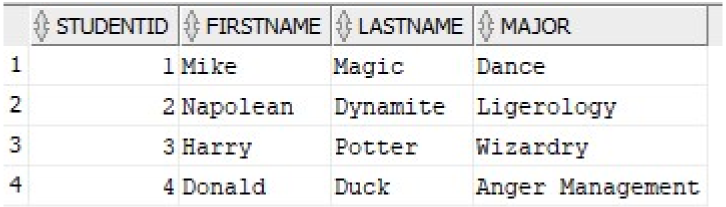
Check to see table updated.

Fig 7o
Code:



```
SELECT * FROM admin.User1Data;
```

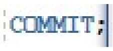
Fig 7p
Output:



	STUDENTID	FIRSTNAME	LASTNAME	MAJOR
1	1	Mike	Magic	Dance
2	2	Napolean	Dynamite	Ligerology
3	3	Harry	Potter	Wizardry
4	4	Donald	Duck	Anger Management

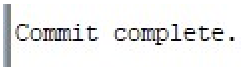
Run Commit statement to ensure changes to table remain after session ends.

Fig 7q
Code:



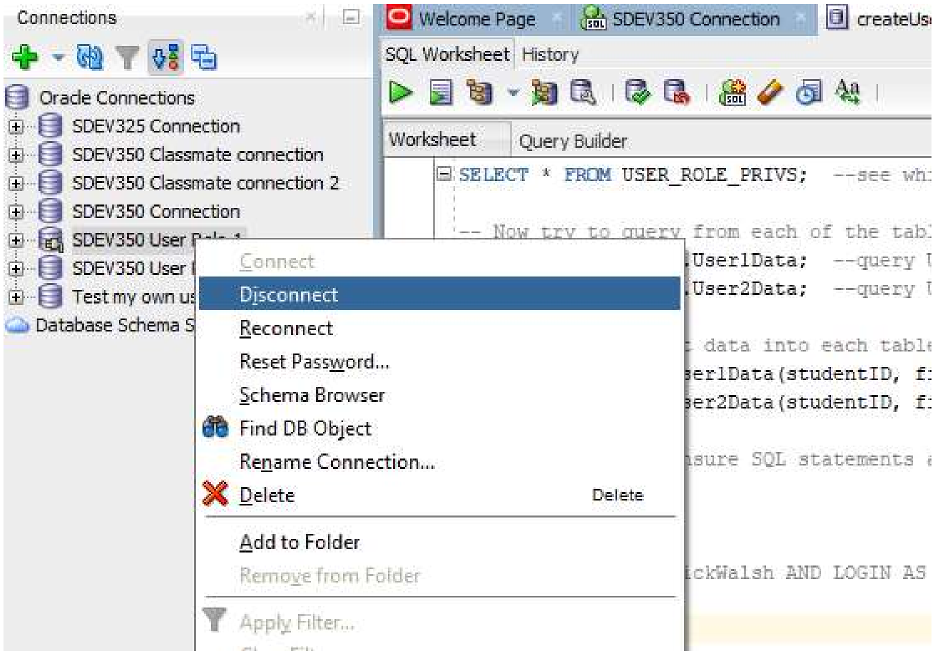
```
COMMIT;
```

Fig 7r
Output:



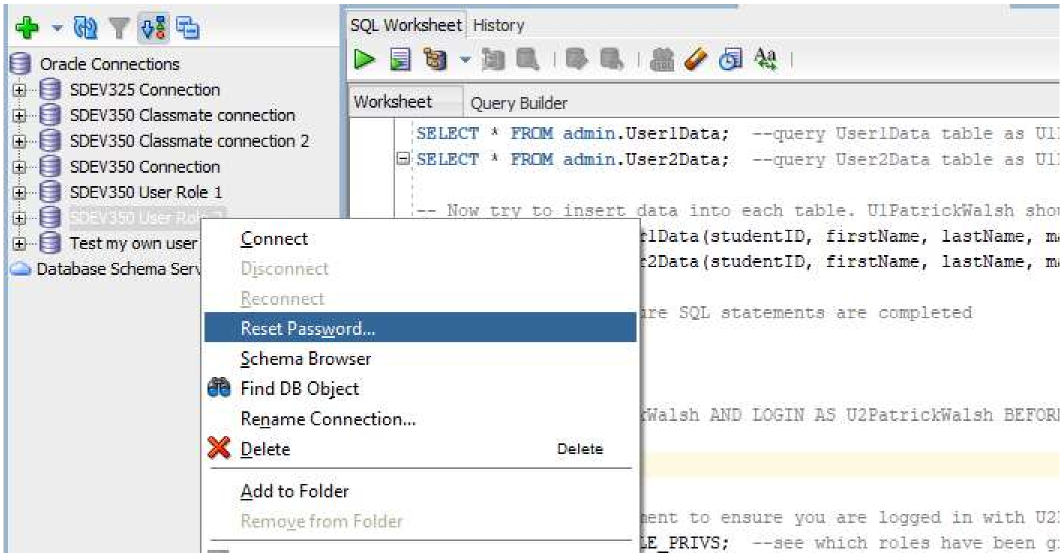
```
Commit complete.
```

Fig 7s
LOGOUT OF U1PatrickWalsh AND LOGIN AS U2PatrickWalsh BEFORE EXECUTING NEXT STATEMENTS



```
SELECT * FROM USER_ROLE_PRIVS;   --see wh

-- Now try to query from each of the tabl
.User1Data;   --query U
.User2Data;   --query U

 data into each table
erlData(studentID, f:
er2Data(studentID, f:

sure SQL statements a

ckWalsh AND LOGIN AS
```

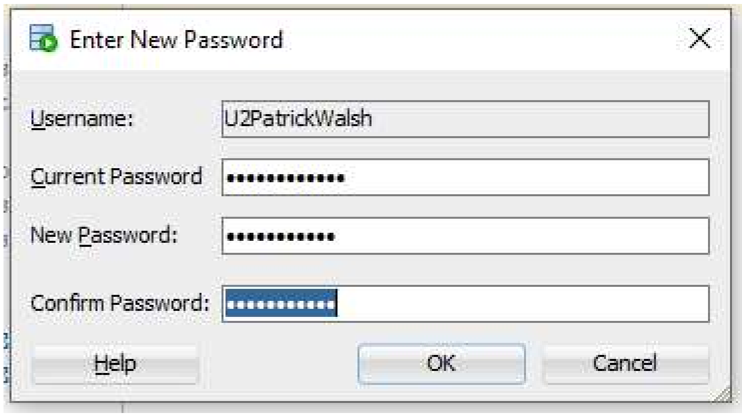
Fig 7t
Then reconnect to database as U2PatrickWalsh. Reset password upon first login:



```
SELECT * FROM admin.User1Data;   --query User1Data table as U1
SELECT * FROM admin.User2Data;   --query User2Data table as U1

-- Now try to insert data into each table. U1PatrickWalsh shou
1Data(studentID, firstName, lastName, m
2Data(studentID, firstName, lastName, m

re SQL statements are completed

Walsh AND LOGIN AS U2PatrickWalsh BEFOR

ent to ensure you are logged in with U2
E_PRIVS;   --see which roles have been g
```

Enter current password = ‘TestPass123!’
New password = ‘NewPass123!’
Confirm new password = ‘NewPass123!’

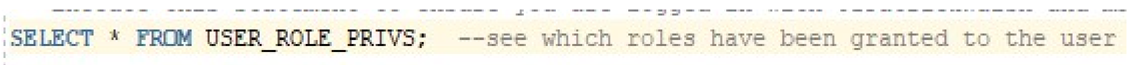
Fig 7u



Username: U2PatrickWalsh
Current Password •••••••••••••
New Password: ••••••••••••
Confirm Password: ••••••••••••

Check user is logged in as U2PatrickWalsh and that you see the appropriate privileges:

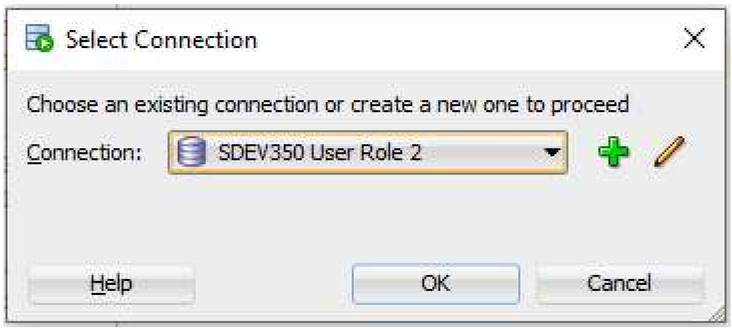
Fig 7v
Code:



```
SELECT * FROM USER_ROLE_PRIVS;  --see which roles have been granted to the user
```

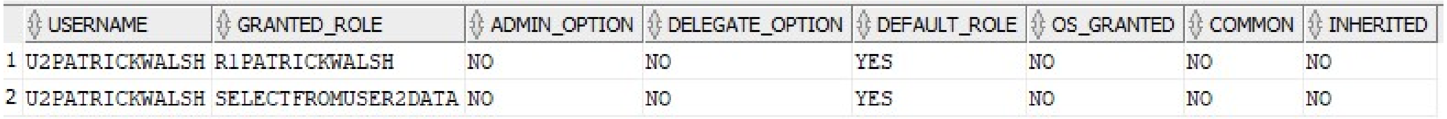
Select connection that uses U1PatrickWalsh user:

Fig 7w



Choose an existing connection or create a new one to proceed
Connection: SDEV350 User Role 2

Fig 7x
Output:

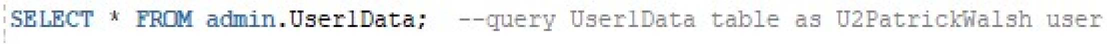


	USERNAME	GRANTED_ROLE	ADMIN_OPTION	DELEGATE_OPTION	DEFAULT_ROLE	OS_GRANTED	COMMON	INHERITED
1	U2PATRICKWALSH	R1PATRICKWALSH	NO	NO	YES	NO	NO	NO
2	U2PATRICKWALSH	SELECTFROMUSER2DATA	NO	NO	YES	NO	NO	NO

Now try to query from each of the tables. U2PatrickWalsh should be able to query User1Data AND User2Data:

Fig 7y

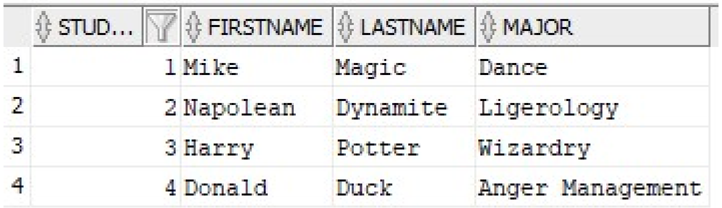
Code(1):



```
SELECT * FROM admin.User1Data;  --query User1Data table as U2PatrickWalsh user
```

Fig 7z

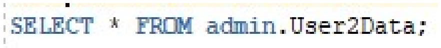
Output(1):



	STUD...	FIRSTNAME	LASTNAME	MAJOR
1	1	Mike	Magic	Dance
2	2	Napolean	Dynamite	Ligerology
3	3	Harry	Potter	Wizardry
4	4	Donald	Duck	Anger Management

Fig 7a.a

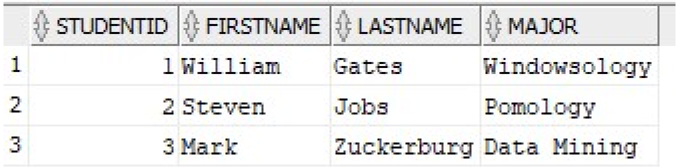
Code(2):



```
SELECT * FROM admin.User2Data;
```

Fig 7a.b

Output(2):

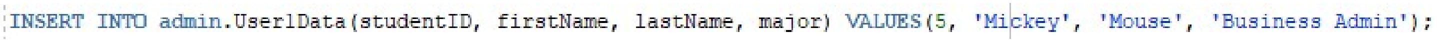


	STUDENTID	FIRSTNAME	LASTNAME	MAJOR
1	1	William	Gates	Windowsology
2	2	Steven	Jobs	Pomology
3	3	Mark	Zuckerburg	Data Mining

Now try to insert data into each table. U2PatrickWalsh should NOT be able to insert into User1Data or User2Data:

Fig 7a.c

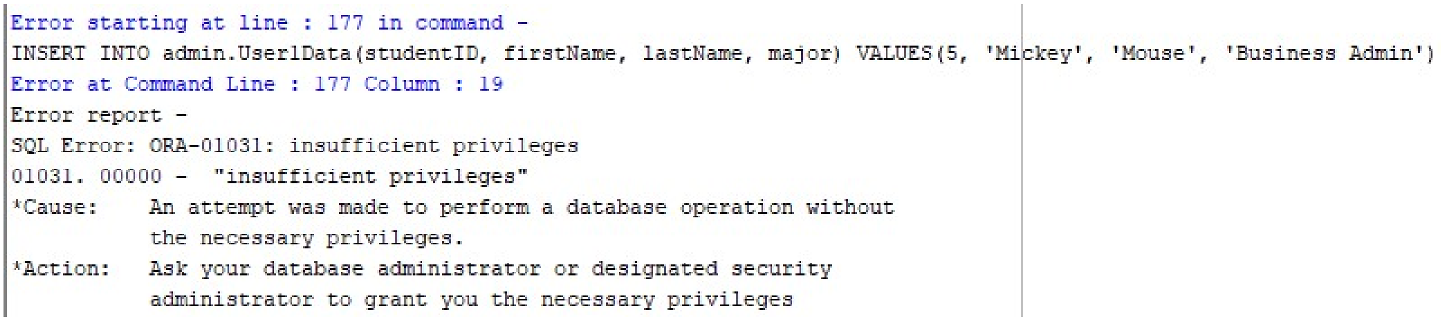
Code(1):



```
INSERT INTO admin.User1Data(studentID, firstName, lastName, major) VALUES(5, 'Mickey', 'Mouse', 'Business Admin');
```

Fig 7a.d

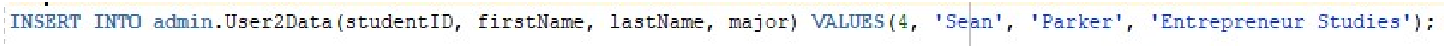
Output(1):



```
Error starting at line : 177 in command -
INSERT INTO admin.User1Data(studentID, firstName, lastName, major) VALUES(5, 'Mickey', 'Mouse', 'Business Admin')
Error at Command Line : 177 Column : 19
Error report -
SQL Error: ORA-01031: insufficient privileges
01031. 00000 -  "insufficient privileges"
*Cause:    An attempt was made to perform a database operation without
           the necessary privileges.
*Action:   Ask your database administrator or designated security
           administrator to grant you the necessary privileges
```

Fig 7a.e

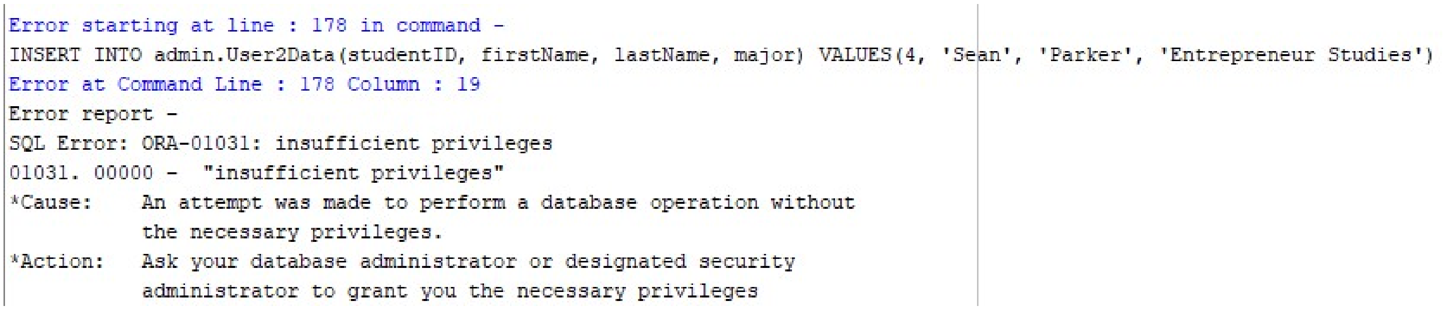
Code(2):



```
INSERT INTO admin.User2Data(studentID, firstName, lastName, major) VALUES(4, 'Sean', 'Parker', 'Entrepreneur Studies');
```

Fig 7a.f

Output(2):



```
Error starting at line : 178 in command -
INSERT INTO admin.User2Data(studentID, firstName, lastName, major) VALUES(4, 'Sean', 'Parker', 'Entrepreneur Studies')
Error at Command Line : 178 Column : 19
Error report -
SQL Error: ORA-01031: insufficient privileges
01031. 00000 -  "insufficient privileges"
*Cause:    An attempt was made to perform a database operation without
           the necessary privileges.
*Action:   Ask your database administrator or designated security
           administrator to grant you the necessary privileges
```

END OF LAB