CNIT487 Database Administration Project

Noah Zhou - Fall 2024

Professor Romila Pradhan

Noah Zhou

Phase 1: Estimate storage requirements for a production Course Enrollment database using the Excel Template provided.

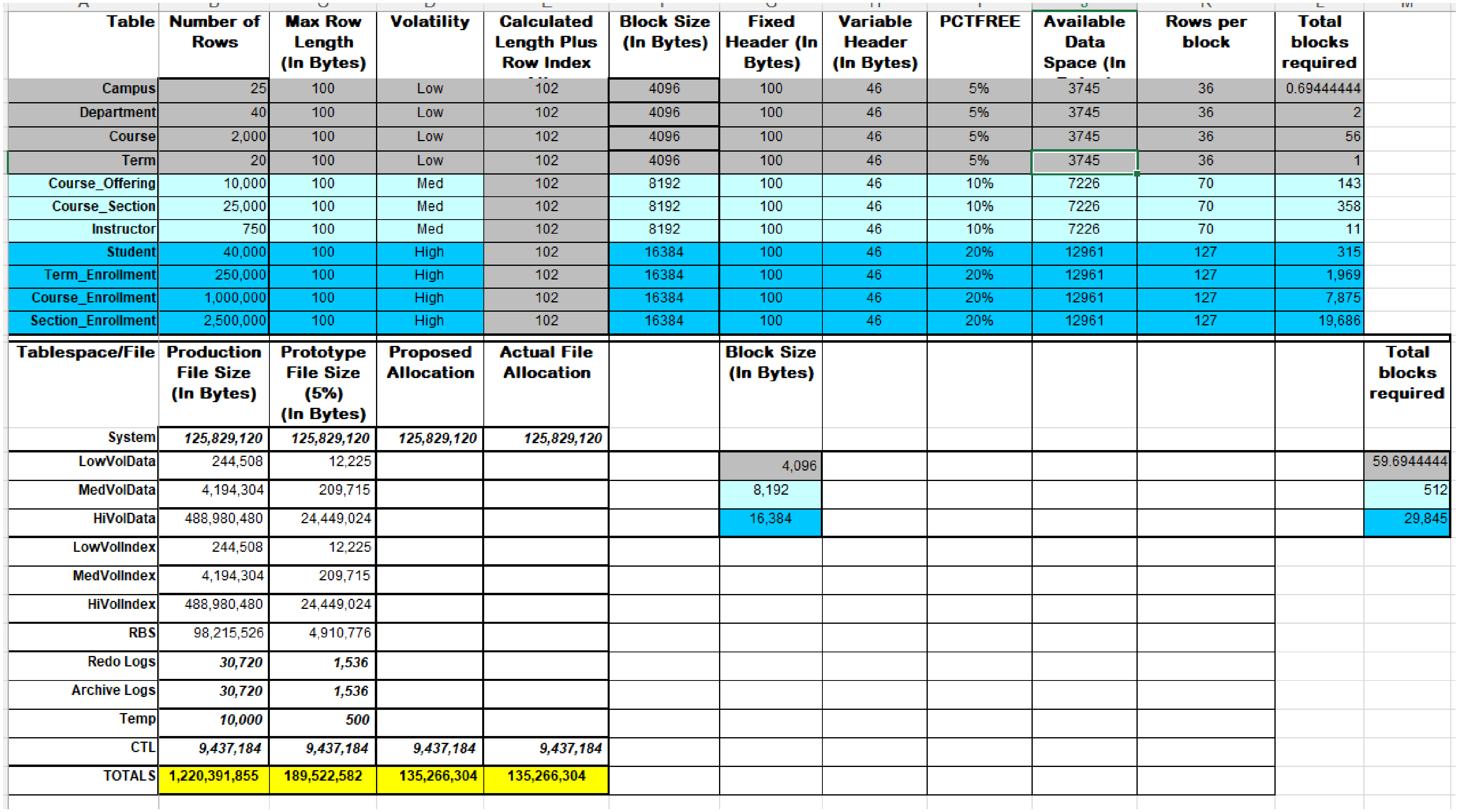


Table	Number of Rows	Max Row Length (In Bytes)	Volatility	Calculated Length Plus Row Index	Block Size (In Bytes)	Fixed Header (In Bytes)	Variable Header (In Bytes)	PCTFREE	Available Data Space (In	Rows per block	Total blocks required
Campus	25	100	Low	102	4096	100	46	5%	3745	36	0.69444444
Department	40	100	Low	102	4096	100	46	5%	3745	36	2
Course	2,000	100	Low	102	4096	100	46	5%	3745	36	56
Term	20	100	Low	102	4096	100	46	5%	3745	36	1
Course_Offering	10,000	100	Med	102	8192	100	46	10%	7226	70	143
Course_Section	25,000	100	Med	102	8192	100	46	10%	7226	70	358
Instructor	750	100	Med	102	8192	100	46	10%	7226	70	11
Student	40,000	100	High	102	16384	100	46	20%	12961	127	315
Term_Enrollment	250,000	100	High	102	16384	100	46	20%	12961	127	1,969
Course_Enrollment	1,000,000	100	High	102	16384	100	46	20%	12961	127	7,875
Section_Enrollment	2,500,000	100	High	102	16384	100	46	20%	12961	127	19,686

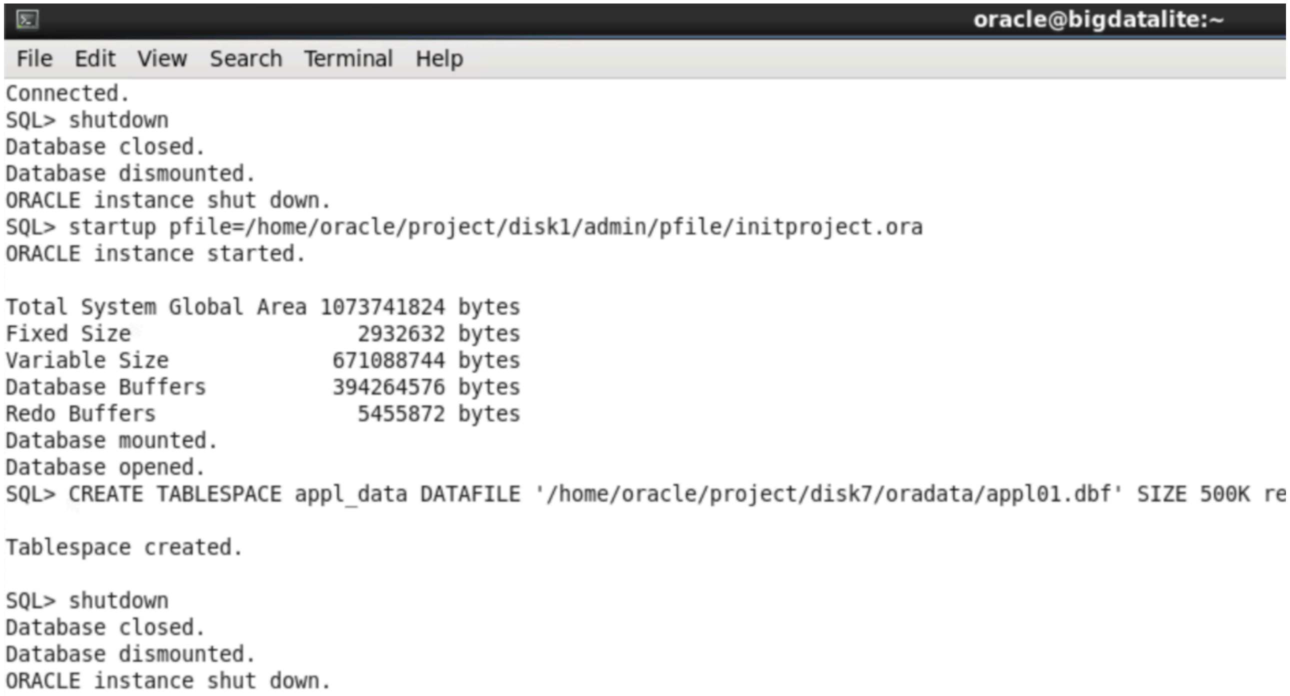
Tablespace/File	Production File Size (In Bytes)	Prototype File Size (5%) (In Bytes)	Proposed Allocation	Actual File Allocation		Block Size (In Bytes)						Total blocks required
System	*125,829,120*	*125,829,120*	*125,829,120*	*125,829,120*								
LowVolData	244,508	12,225				4,096						59.6944444
MedVolData	4,194,304	209,715				8,192						512
HiVolData	488,980,480	24,449,024				16,384						29,845
LowVolIndex	244,508	12,225										
MedVolIndex	4,194,304	209,715										
HiVolIndex	488,980,480	24,449,024										
RBS	98,215,526	4,910,776										
Redo Logs	*30,720*	*1,536*										
Archive Logs	*30,720*	*1,536*										
Temp	*10,000*	*500*										
CTL	*9,437,184*	*9,437,184*	*9,437,184*	*9,437,184*								
TOTALS	1,220,391,855	189,522,582	135,266,304	135,266,304								

This is the Excel spreadsheet used in Phase 1 of the database project which depicts the max row length, volatility, and block sizes of each table. I also added in the block sizes of each tablespace: LowVolData, MedVolData, and HighVolData. These tablespaces and their custom bock sizes will be created in Phase 2 of the project.

Noah Zhou

Phase 2:

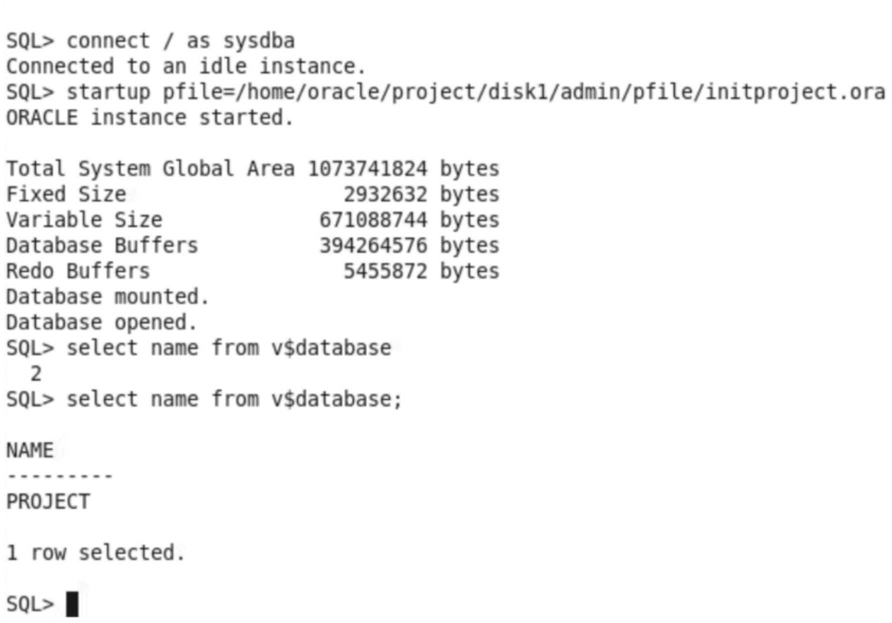
Create a prototype of the production database that can be used for application development.



```
Connected.
SQL> shutdown
Database closed.
Database dismounted.
ORACLE instance shut down.
SQL> startup pfile=/home/oracle/project/disk1/admin/pfile/initproject.ora
ORACLE instance started.

Total System Global Area 1073741824 bytes
Fixed Size                  2932632 bytes
Variable Size             671088744 bytes
Database Buffers          394264576 bytes
Redo Buffers                5455872 bytes
Database mounted.
Database opened.
SQL> CREATE TABLESPACE appl_data DATAFILE '/home/oracle/project/disk7/oradata/appl01.dbf' SIZE 500K re

Tablespace created.

SQL> shutdown
Database closed.
Database dismounted.
ORACLE instance shut down.
```

To start Phase 2, I created the prototype project database by replicating and modifying the given lab database files and folders. I then took the lab parameter initialization file `initcit487.ora` and named this new file `initproject.ora`.

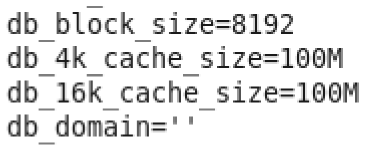
Noah Zhou



```
SQL> connect / as sysdba
Connected to an idle instance.
SQL> startup pfile=/home/oracle/project/disk1/admin/pfile/initproject.ora
ORACLE instance started.

Total System Global Area 1073741824 bytes
Fixed Size                  2932632 bytes
Variable Size             671088744 bytes
Database Buffers          394264576 bytes
Redo Buffers                5455872 bytes
Database mounted.
Database opened.
SQL> select name from v$database
  2
SQL> select name from v$database;

NAME
---------
PROJECT

1 row selected.

SQL> █
```

To verify that the database existed after starting it up using the new parameter file, I queried the `v$database` view which shows the database in which I am currently connected to.

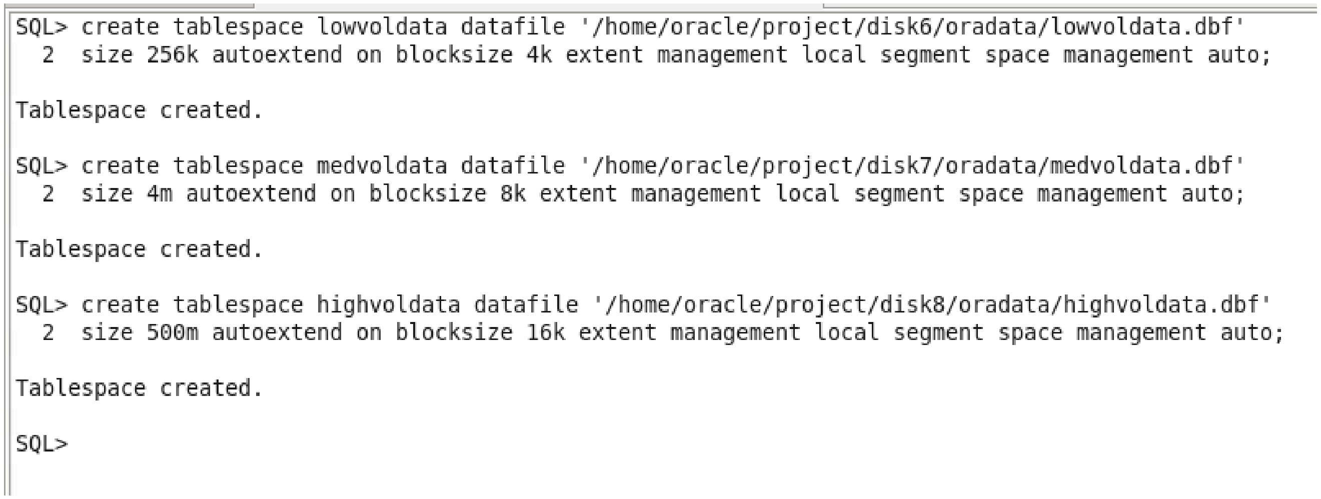
Create tables and indexes with appropriate tablespace, PCTFREE and storage allocations.



```
db_block_size=8192
db_4k_cache_size=100M
db_16k_cache_size=100M
db_domain=''
```

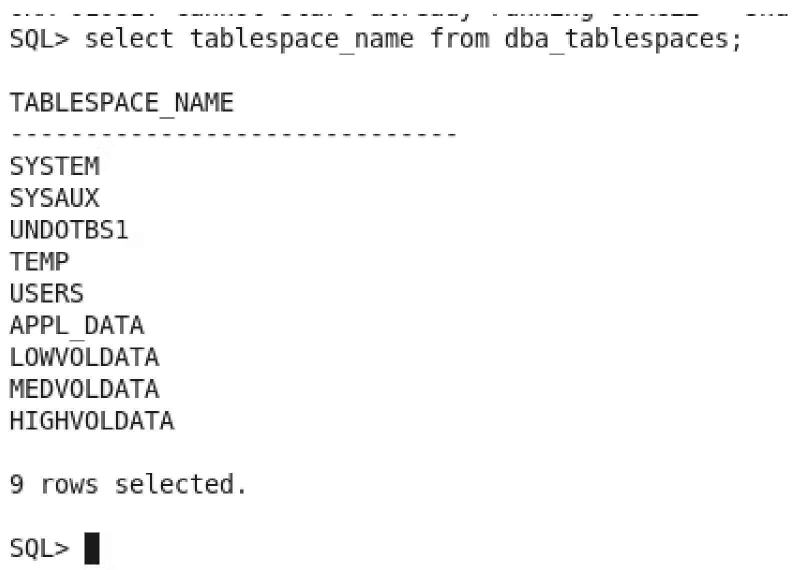
In order to create tablespaces with custom block sizes, it was necessary for me to edit the initialization parameter file initproject.ora and add min values for the cache sizes. As you can see, I added in `db_4k_cache_size=100M` and `db_16k_cache_size=100M`. These two additions to the parameter file allowed custom tablespace block sizes of `4k` and `16k` for the respective tablespaces. The reason why there is no `8k` cache size is because the default block size for tablespaces in Oracle is `8k`.

Noah Zhou



```
SQL> create tablespace lowvoldata datafile '/home/oracle/project/disk6/oradata/lowvoldata.dbf'
  2  size 256k autoextend on blocksize 4k extent management local segment space management auto;

Tablespace created.

SQL> create tablespace medvoldata datafile '/home/oracle/project/disk7/oradata/medvoldata.dbf'
  2  size 4m autoextend on blocksize 8k extent management local segment space management auto;

Tablespace created.

SQL> create tablespace highvoldata datafile '/home/oracle/project/disk8/oradata/highvoldata.dbf'
  2  size 500m autoextend on blocksize 16k extent management local segment space management auto;

Tablespace created.

SQL>
```

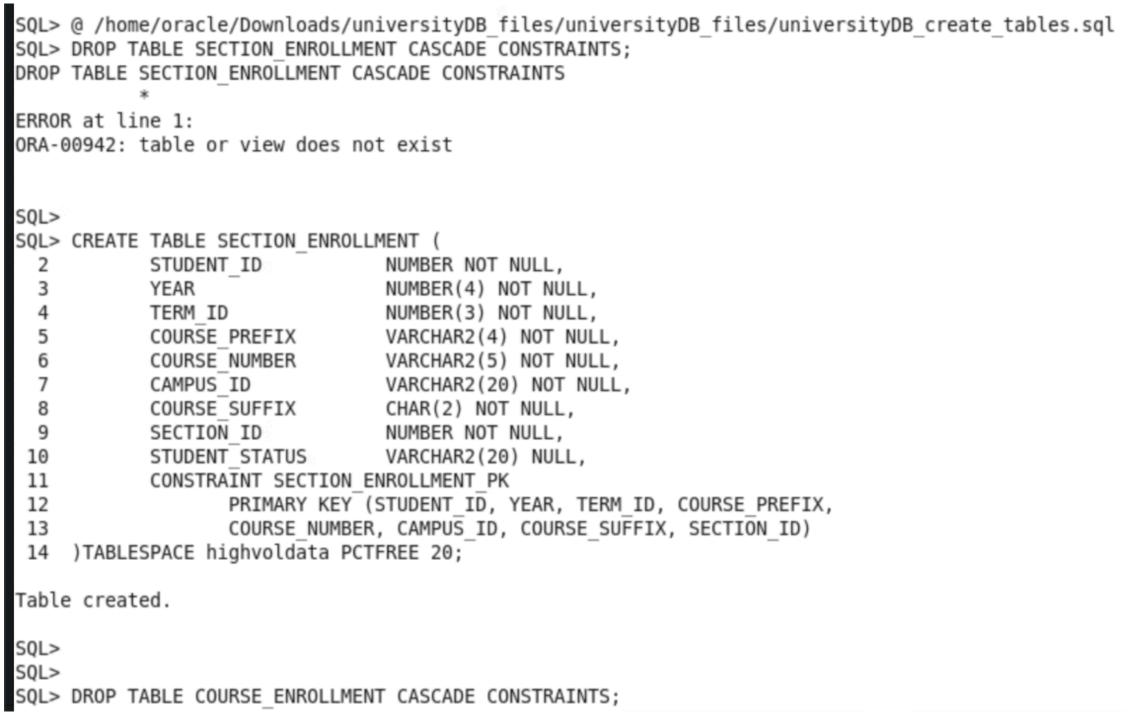
Now that I was able to create tablespaces with custom block sizes, I created the three required tablespaces: LowVolData, MedVolData, and HighVolData using the `create tablespace ‘name’, datafile ‘datafile/location’ size ‘tablespace_size’, autoextend on, blocksize ‘custom_block_size’ extent management local segment space management auto;` command.



```
SQL> select tablespace_name from dba_tablespaces;

TABLESPACE_NAME
------------------------------
SYSTEM
SYSAUX
UNDOTBS1
TEMP
USERS
APPL_DATA
LOWVOLDATA
MEDVOLDATA
HIGHVOLDATA

9 rows selected.

SQL> 
```

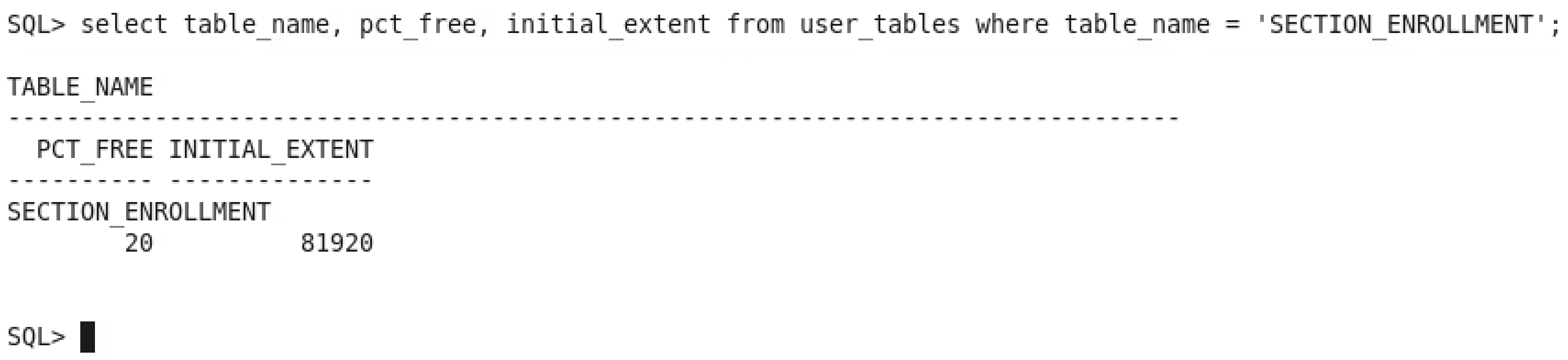
To verify that I had successfully created the tablespaces, I queried the `dba_tablespaces` table. Here we can see that the LowVolData, MedVolData, and HighVolData tablespaces have been successfully created.

Noah Zhou



```
SQL> @ /home/oracle/Downloads/universityDB_files/universityDB_files/universityDB_create_tables.sql
SQL> DROP TABLE SECTION_ENROLLMENT CASCADE CONSTRAINTS;
DROP TABLE SECTION_ENROLLMENT CASCADE CONSTRAINTS
           *
ERROR at line 1:
ORA-00942: table or view does not exist


SQL>
SQL> CREATE TABLE SECTION_ENROLLMENT (
  2         STUDENT_ID          NUMBER NOT NULL,
  3         YEAR                NUMBER(4) NOT NULL,
  4         TERM_ID             NUMBER(3) NOT NULL,
  5         COURSE_PREFIX       VARCHAR2(4) NOT NULL,
  6         COURSE_NUMBER       VARCHAR2(5) NOT NULL,
  7         CAMPUS_ID           VARCHAR2(20) NOT NULL,
  8         COURSE_SUFFIX       CHAR(2) NOT NULL,
  9         SECTION_ID          NUMBER NOT NULL,
 10         STUDENT_STATUS      VARCHAR2(20) NULL,
 11         CONSTRAINT SECTION_ENROLLMENT_PK
 12                PRIMARY KEY (STUDENT_ID, YEAR, TERM_ID, COURSE_PREFIX,
 13                COURSE_NUMBER, CAMPUS_ID, COURSE_SUFFIX, SECTION_ID)
 14  )TABLESPACE highvoldata PCTFREE 20;

Table created.

SQL>
SQL>
SQL> DROP TABLE COURSE_ENROLLMENT CASCADE CONSTRAINTS;
```

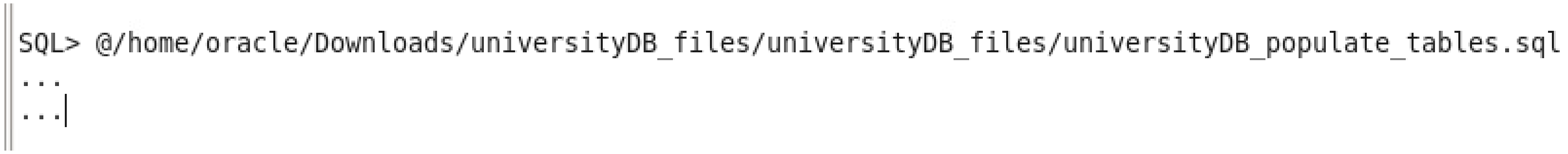
The next step was to create the tables that are supposed to be part of the database. I first edited the provided `universityDB_create_tables.sql` file to include the tablespaces that each table would go into in addition to the `PCTFREE` allocation for each table. I then saved and ran the file which created all of the necessary tables.



```
SQL> select table_name, pct_free, initial_extent from user_tables where table_name = 'SECTION_ENROLLMENT';

TABLE_NAME
--------------------------------------------------------------------------------
  PCT_FREE INITIAL_EXTENT
---------- --------------
SECTION_ENROLLMENT
        20          81920


SQL> 
```

I then queried the `user_tables` table and specified the table name as `SECTION_ENROLLMENT`. We can see that the table exists and has a `PCTFREE` allocation.

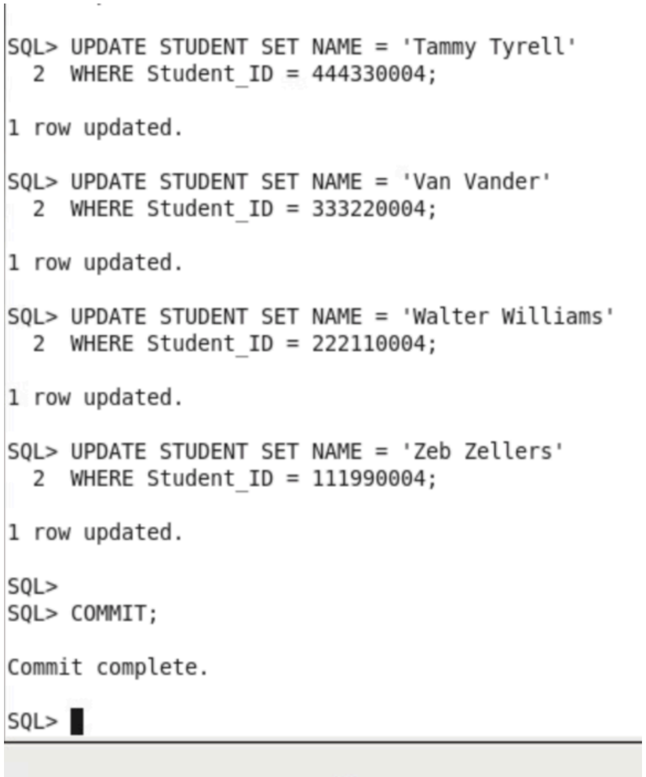
Populate the databases using the script files provided and SQL*Loader or other utility/technique of your choice, then test the databases to make sure it is usable.



```
SQL> @/home/oracle/Downloads/universityDB_files/universityDB_files/universityDB_populate_tables.sql
...
...
```

To populate the database and tables, I ran the given `universityDB_populate_tables.sql` file which populated all of the tables I had previously created.

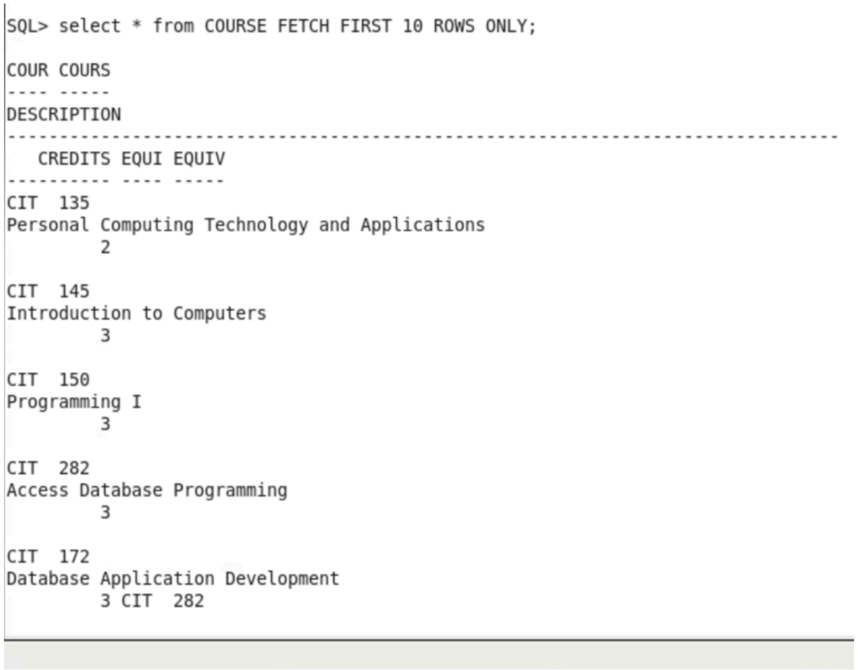
Noah Zhou



```
SQL> UPDATE STUDENT SET NAME = 'Tammy Tyrell'
  2  WHERE Student_ID = 444330004;

1 row updated.

SQL> UPDATE STUDENT SET NAME = 'Van Vander'
  2  WHERE Student_ID = 333220004;

1 row updated.

SQL> UPDATE STUDENT SET NAME = 'Walter Williams'
  2  WHERE Student_ID = 222110004;

1 row updated.

SQL> UPDATE STUDENT SET NAME = 'Zeb Zellers'
  2  WHERE Student_ID = 111990004;

1 row updated.

SQL>
SQL> COMMIT;

Commit complete.

SQL> █
```

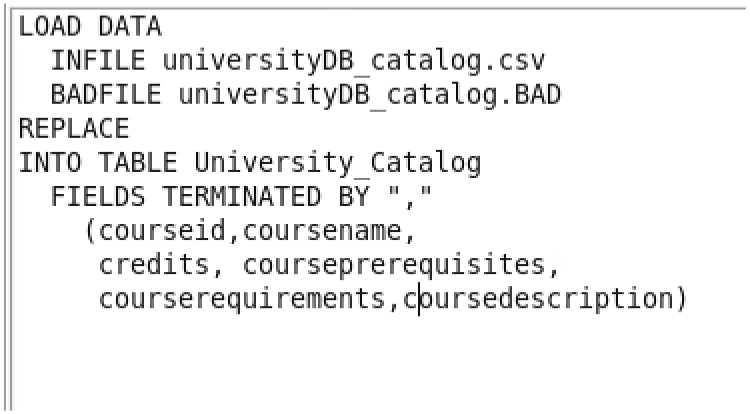
As we can see, the sql file was successfully run and the data was committed to the database.

Noah Zhou



```
SQL> select * from COURSE FETCH FIRST 10 ROWS ONLY;

COUR COURS
---- -----
DESCRIPTION
--------------------------------------------------------------------------------
   CREDITS EQUI EQUIV
---------- ---- -----
CIT  135
Personal Computing Technology and Applications
         2

CIT  145
Introduction to Computers
         3

CIT  150
Programming I
         3

CIT  282
Access Database Programming
         3

CIT  172
Database Application Development
         3 CIT  282
```

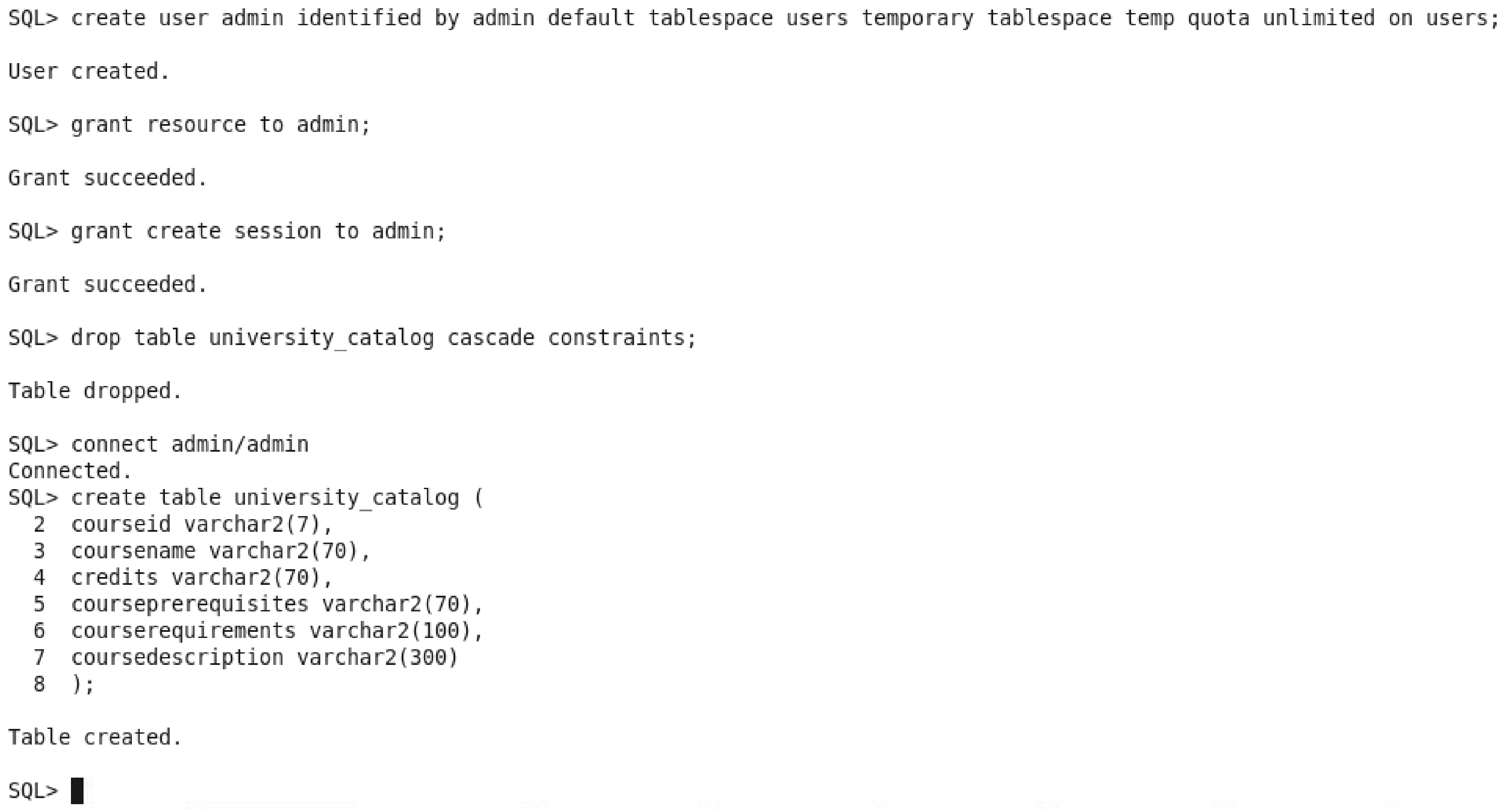
To verify that the data that was inserted truly existed, I queried the `COURSE` table to show that data was successfully inserted.



```
LOAD DATA
  INFILE universityDB_catalog.csv
  BADFILE universityDB_catalog.BAD
REPLACE
INTO TABLE University_Catalog
  FIELDS TERMINATED BY ","
    (courseid,coursename,
     credits, courseprerequisites,
     courserequirements,coursedescription)
```

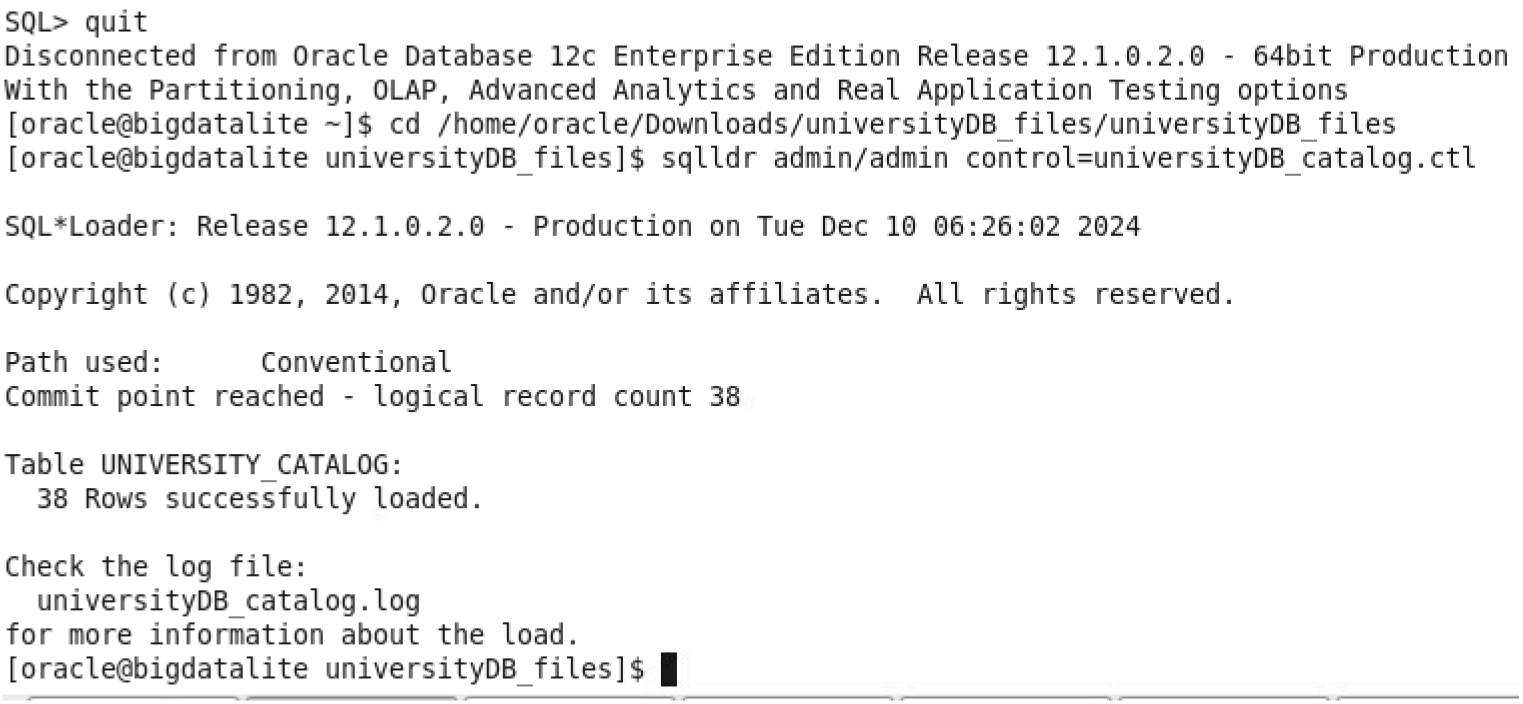
Noah Zhou

In order to use SQL*Loader to import data from the provided `universityDB_catalog.csv` file, I first had to create a control file as seen above. I created this control file by using the given control file in Lab 4 and labeled it as `universityDB_catalog.ctl`. The control file specified where the data would come from and to what table the data would be loaded into.



```
SQL> create user admin identified by admin default tablespace users temporary tablespace temp quota unlimited on users;

User created.

SQL> grant resource to admin;

Grant succeeded.

SQL> grant create session to admin;

Grant succeeded.

SQL> drop table university_catalog cascade constraints;

Table dropped.

SQL> connect admin/admin
Connected.
SQL> create table university_catalog (
  2  courseid varchar2(7),
  3  coursename varchar2(70),
  4  credits varchar2(70),
  5  courseprerequisites varchar2(70),
  6  courserequirements varchar2(100),
  7  coursedescription varchar2(300)
  8  );

Table created.

SQL> 
```

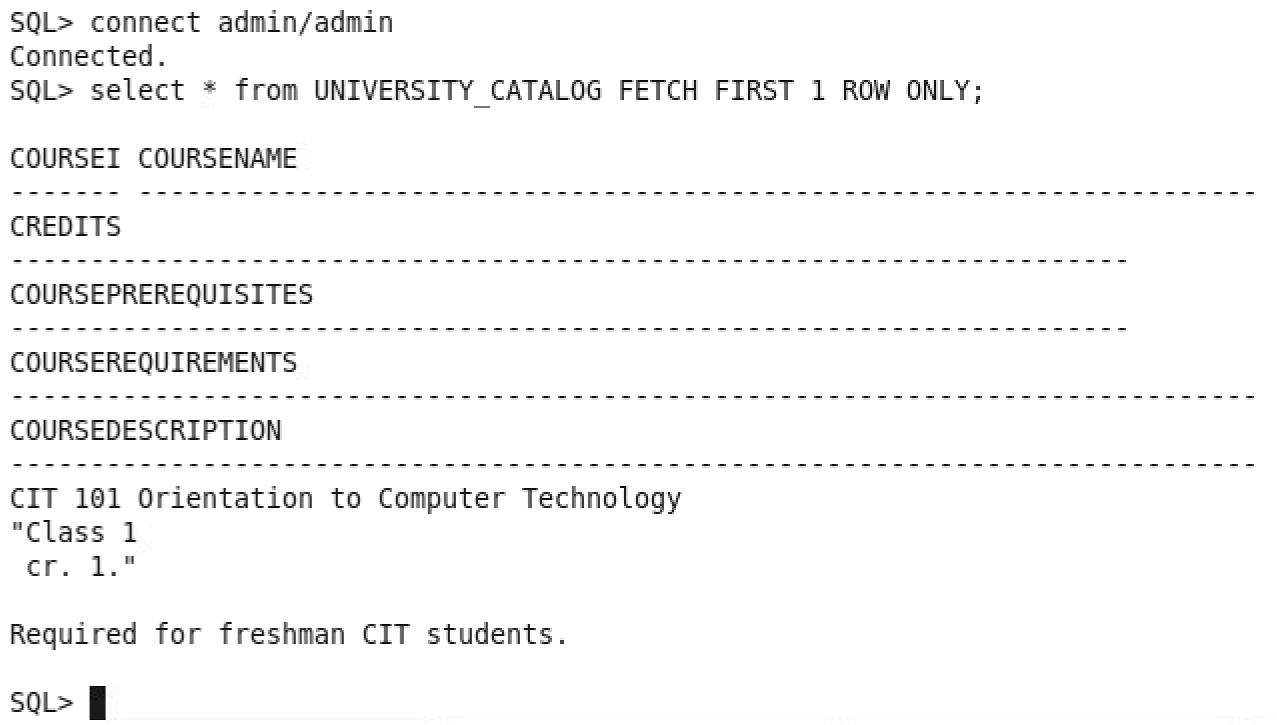
After creating the control file, I created a user named admin and gave them permissions to create tables and to create a session. I then logged in as admin and created the table for the universityDB_catalog.csv file with column names and datatypes that matched each column in the CSV file. The table was named university_catalog.



```
SQL> quit
Disconnected from Oracle Database 12c Enterprise Edition Release 12.1.0.2.0 - 64bit Production
With the Partitioning, OLAP, Advanced Analytics and Real Application Testing options
[oracle@bigdatalite ~]$ cd /home/oracle/Downloads/universityDB_files/universityDB_files
[oracle@bigdatalite universityDB_files]$ sqlldr admin/admin control=universityDB_catalog.ctl

SQL*Loader: Release 12.1.0.2.0 - Production on Tue Dec 10 06:26:02 2024

Copyright (c) 1982, 2014, Oracle and/or its affiliates.  All rights reserved.

Path used:      Conventional
Commit point reached - logical record count 38

Table UNIVERSITY_CATALOG:
  38 Rows successfully loaded.

Check the log file:
  universityDB_catalog.log
for more information about the load.
[oracle@bigdatalite universityDB_files]$ 
```

After creating the user, I used SQL*Loader to load the CSV file by running the control file. As we can see here, 38 rows of data were successfully loaded.

Noah Zhou

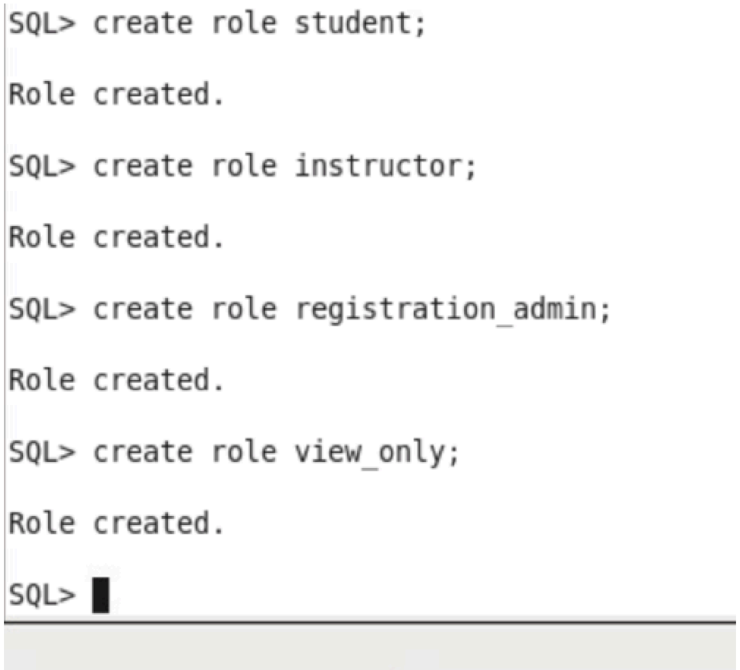


```
SQL> connect admin/admin
Connected.
SQL> select * from UNIVERSITY_CATALOG FETCH FIRST 1 ROW ONLY;

COURSEI COURSENAME
------- ----------------------------------------------------------------------
CREDITS
----------------------------------------------------------------------
COURSEPREREQUISITES
----------------------------------------------------------------------
COURSEREQUIREMENTS
------------------------------------------------------------------------------
COURSEDESCRIPTION
------------------------------------------------------------------------------
CIT 101 Orientation to Computer Technology
"Class 1
 cr. 1."

Required for freshman CIT students.

SQL> █
```

To verify that the data did in fact exist, I logged in as the `admin` user and ran a simple select query that successfully printed results from the `UNIVERSITY_CATALOG` table.

Noah Zhou

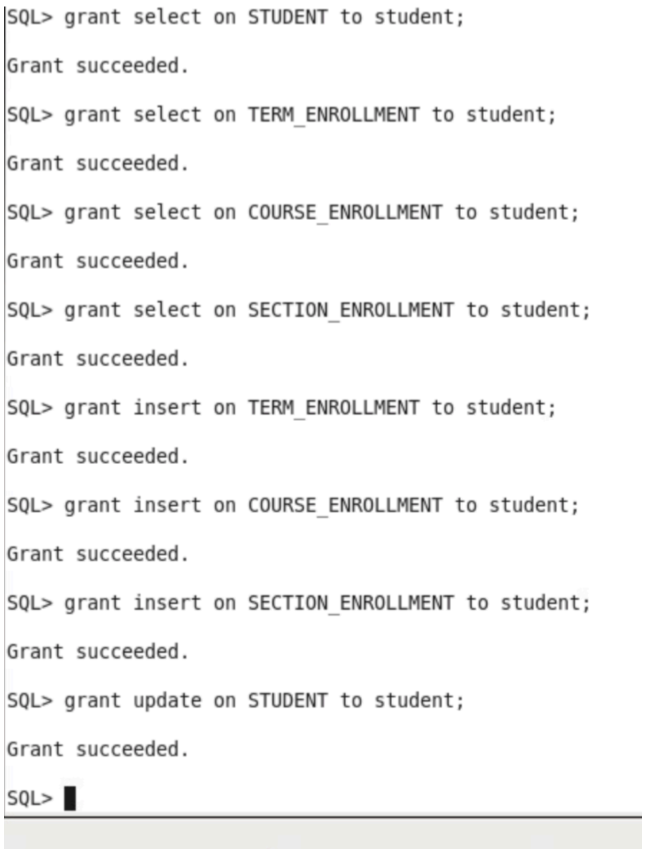
Create development users and roles, then grant privileges as appropriate.



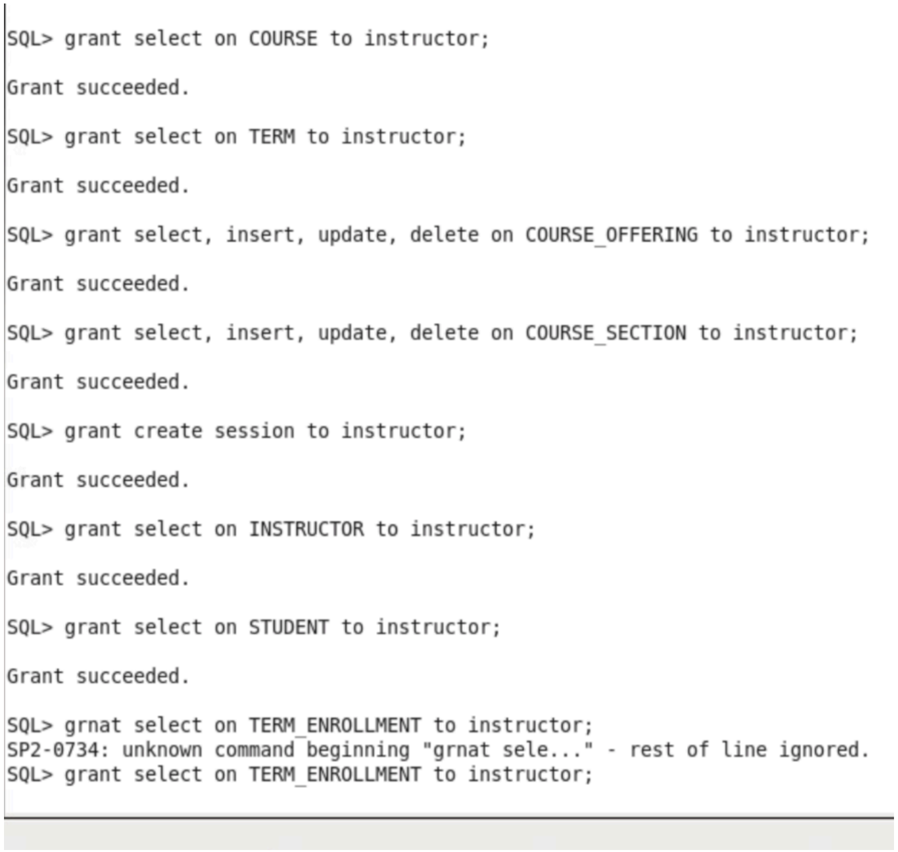
```
SQL> create role student;

Role created.

SQL> create role instructor;

Role created.

SQL> create role registration_admin;

Role created.

SQL> create role view_only;

Role created.

SQL> 
```

I first created 4 separate roles; `student`, `instructor`, `registration_admin`, and `view_only`.

Noah Zhou



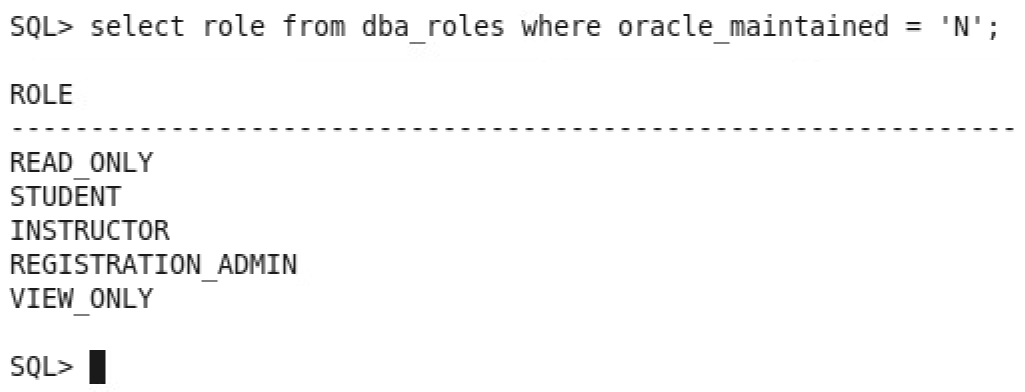
```
SQL> grant select on STUDENT to student;

Grant succeeded.

SQL> grant select on TERM_ENROLLMENT to student;

Grant succeeded.

SQL> grant select on COURSE_ENROLLMENT to student;

Grant succeeded.

SQL> grant select on SECTION_ENROLLMENT to student;

Grant succeeded.

SQL> grant insert on TERM_ENROLLMENT to student;

Grant succeeded.

SQL> grant insert on COURSE_ENROLLMENT to student;

Grant succeeded.

SQL> grant insert on SECTION_ENROLLMENT to student;

Grant succeeded.

SQL> grant update on STUDENT to student;

Grant succeeded.

SQL> 
```



```
SQL> grant select on COURSE to instructor;

Grant succeeded.

SQL> grant select on TERM to instructor;

Grant succeeded.

SQL> grant select, insert, update, delete on COURSE_OFFERING to instructor;

Grant succeeded.

SQL> grant select, insert, update, delete on COURSE_SECTION to instructor;

Grant succeeded.

SQL> grant create session to instructor;

Grant succeeded.

SQL> grant select on INSTRUCTOR to instructor;

Grant succeeded.

SQL> grant select on STUDENT to instructor;

Grant succeeded.

SQL> grnat select on TERM_ENROLLMENT to instructor;
SP2-0734: unknown command beginning "grnat sele..." - rest of line ignored.
SQL> grant select on TERM_ENROLLMENT to instructor;
```

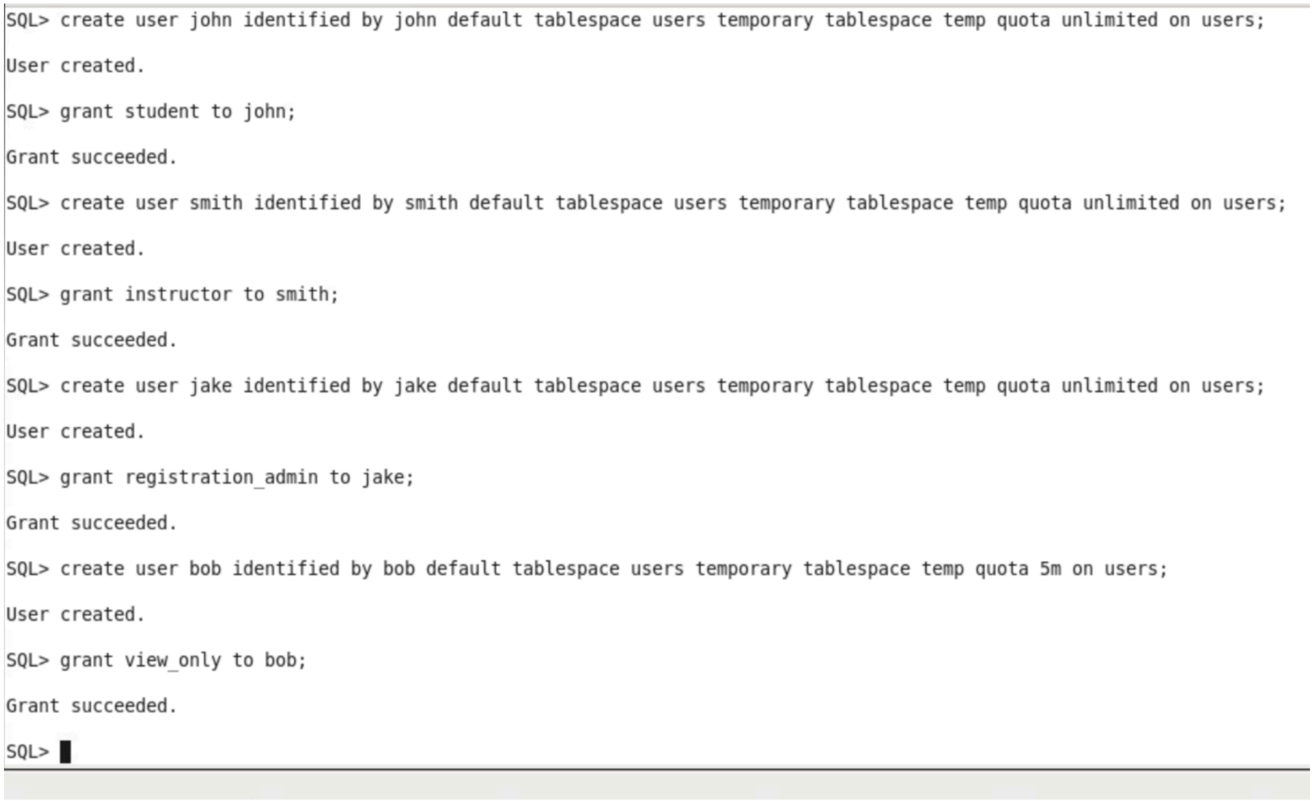
I then granted specific permissions to each role, based on my interpretation of what permissions each role should have.

Noah Zhou



```
SQL> select role from dba_roles where oracle_maintained = 'N';

ROLE
--------------------------------------------------------------
READ_ONLY
STUDENT
INSTRUCTOR
REGISTRATION_ADMIN
VIEW_ONLY

SQL> █
```

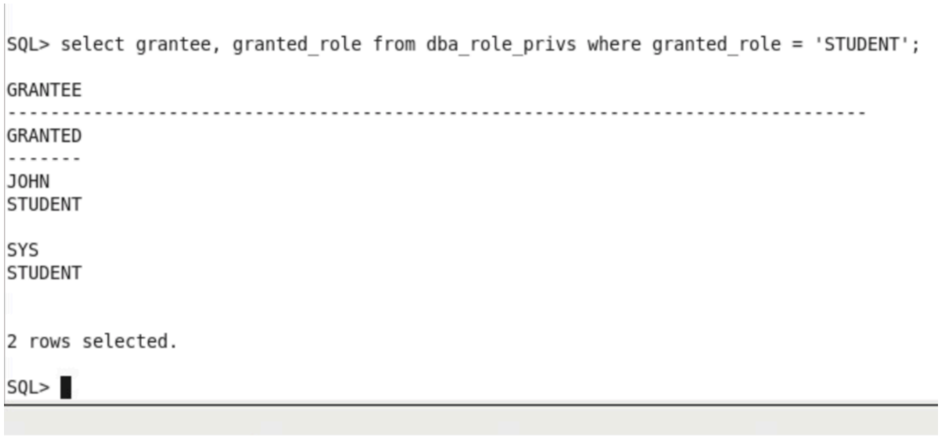
To check and make sure that the roles that I created existed, I queried the `dba_roles table` and specified the parameter that `oracle_mainted=’N’`. This parameter makes it so that only user created roles are shown.



```
SQL> create user john identified by john default tablespace users temporary tablespace temp quota unlimited on users;

User created.

SQL> grant student to john;

Grant succeeded.

SQL> create user smith identified by smith default tablespace users temporary tablespace temp quota unlimited on users;

User created.

SQL> grant instructor to smith;

Grant succeeded.

SQL> create user jake identified by jake default tablespace users temporary tablespace temp quota unlimited on users;

User created.

SQL> grant registration_admin to jake;

Grant succeeded.

SQL> create user bob identified by bob default tablespace users temporary tablespace temp quota 5m on users;

User created.

SQL> grant view_only to bob;

Grant succeeded.

SQL> █
```

Following the verification of roles, I then created sample users and assigned each user to a role.

Noah Zhou

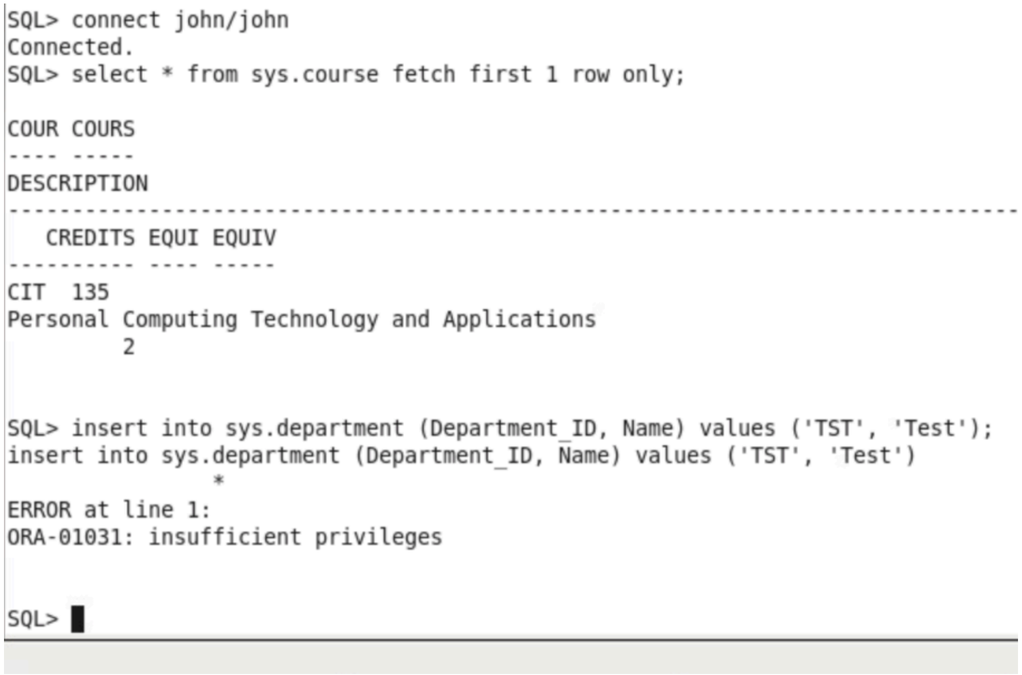


```
SQL> select grantee, granted_role from dba_role_privs where granted_role = 'STUDENT';

GRANTEE
--------------------------------------------------------------------------------
GRANTED
-------
JOHN
STUDENT

SYS
STUDENT


2 rows selected.

SQL> █
```

I then queried the `dba_role_privs` table and specified the role as `Student`. We can see that the `Student` role has a user that was created earlier..

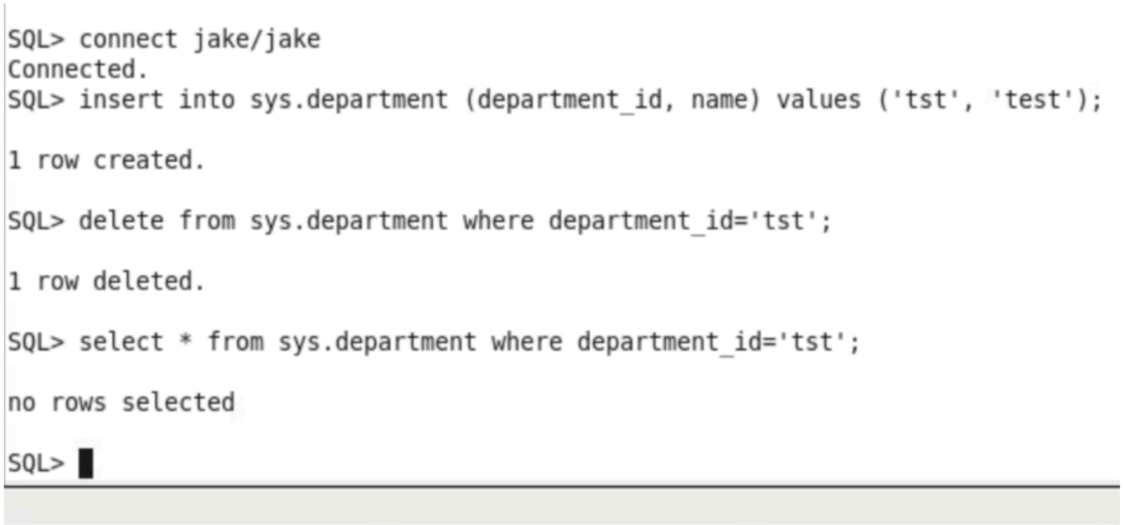
Noah Zhou

Demonstrate scenarios where different user roles attempt to access the database in unauthorized ways.

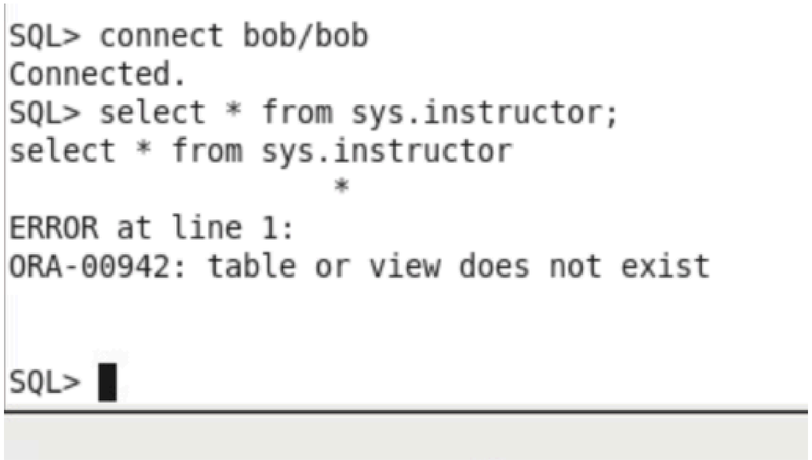


```
SQL> connect john/john
Connected.
SQL> select * from sys.course fetch first 1 row only;

COUR COURS
---- -----
DESCRIPTION
-------------------------------------------------------------------------------
   CREDITS EQUI EQUIV
---------- ---- -----
CIT  135
Personal Computing Technology and Applications
         2


SQL> insert into sys.department (Department_ID, Name) values ('TST', 'Test');
insert into sys.department (Department_ID, Name) values ('TST', 'Test')
                *
ERROR at line 1:
ORA-01031: insufficient privileges


SQL> █
```

Noah Zhou



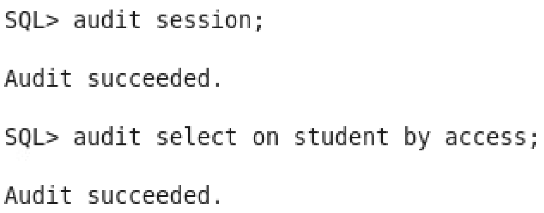
```
SQL> connect jake/jake
Connected.
SQL> insert into sys.department (department_id, name) values ('tst', 'test');

1 row created.

SQL> delete from sys.department where department_id='tst';

1 row deleted.

SQL> select * from sys.department where department_id='tst';

no rows selected

SQL>
```



```
SQL> connect bob/bob
Connected.
SQL> select * from sys.instructor;
select * from sys.instructor
                  *
ERROR at line 1:
ORA-00942: table or view does not exist


SQL>
```

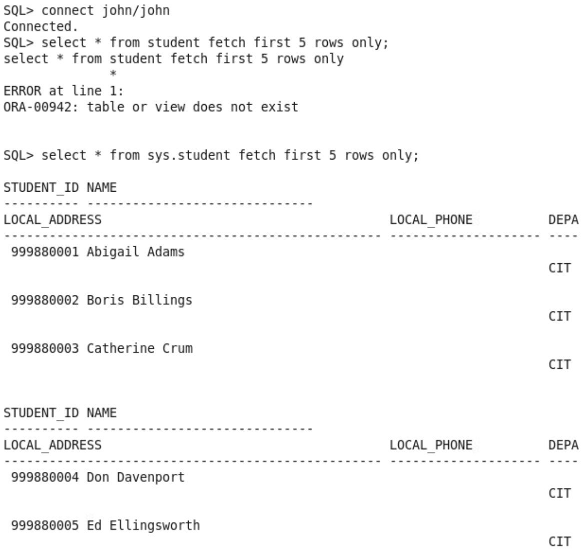
To test various user roles and their permissions, I logged into 3 different users and ran sample queries against the tables in which they did and did not have permissions. We can see that for the users who did not have permissions to access certain tables, the tables just did not exist for them.

Noah Zhou



```
SQL> audit session;

Audit succeeded.

SQL> audit select on student by access;

Audit succeeded.
```

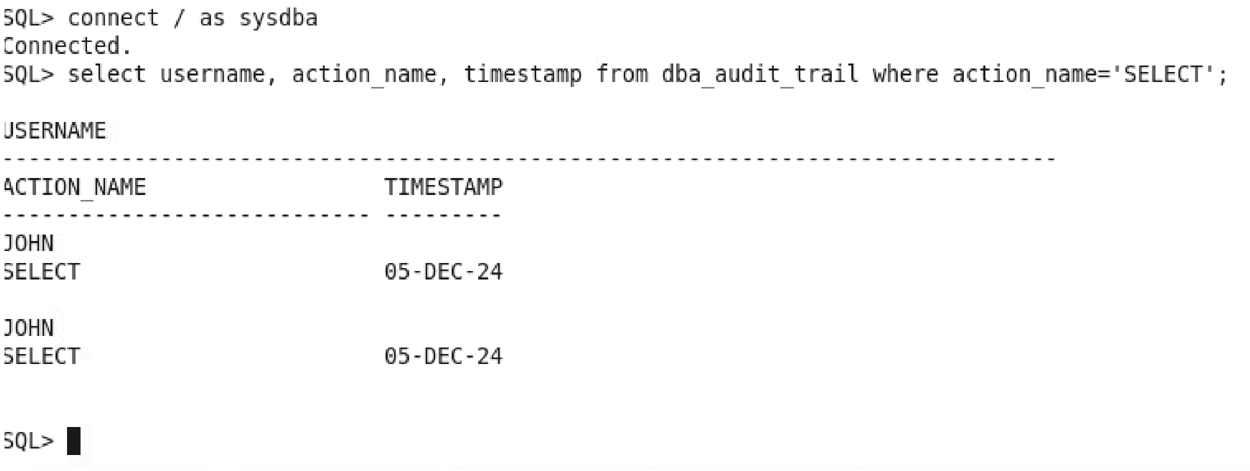
To start auditing, I first ran the `audit session;` command. I then specified auditing on the `Student` table where the action was `SELECT`.



```
SQL> connect john/john
Connected.
SQL> select * from student fetch first 5 rows only;
select * from student fetch first 5 rows only
              *
ERROR at line 1:
ORA-00942: table or view does not exist


SQL> select * from sys.student fetch first 5 rows only;

STUDENT_ID NAME
---------- ------------------------------
LOCAL_ADDRESS                                      LOCAL_PHONE          DEPA
-------------------------------------------------- -------------------- ----
 999880001 Abigail Adams
                                                                        CIT

 999880002 Boris Billings
                                                                        CIT

 999880003 Catherine Crum
                                                                        CIT


STUDENT_ID NAME
---------- ------------------------------
LOCAL_ADDRESS                                      LOCAL_PHONE          DEPA
-------------------------------------------------- -------------------- ----
 999880004 Don Davenport
                                                                        CIT

 999880005 Ed Ellingsworth
                                                                        CIT
```

To test auditing, I first logged in as a user who had access to the `Student` table and ran a simple select query.

Noah Zhou



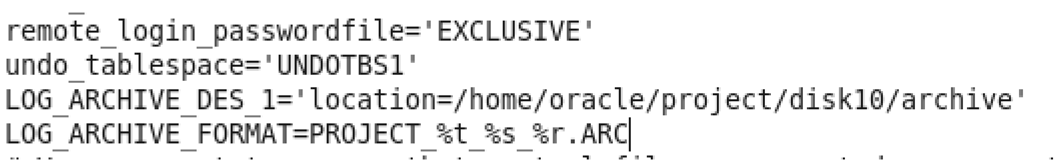
```
SQL> connect / as sysdba
Connected.
SQL> select username, action_name, timestamp from dba_audit_trail where action_name='SELECT';

USERNAME
--------------------------------------------------------------------------------
ACTION_NAME                  TIMESTAMP
---------------------------- ---------
JOHN
SELECT                       05-DEC-24

JOHN
SELECT                       05-DEC-24


SQL> █
```

To check whether the audit was successful, I logged in as the `SYSDBA` and queried the `dba_audit_trail` table and specified the action name as `SELECT`. We can see that the user that I previously logged in as had ran a select statement which demonstrates that the audit was successful.

Noah Zhou

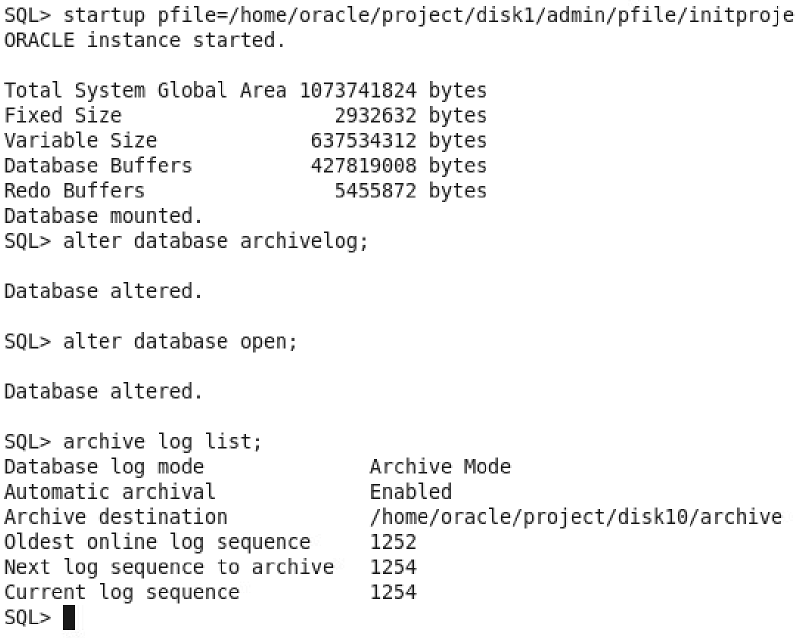
Phase 3:

Develop database backup and recovery procedures.



```
remote_login_passwordfile='EXCLUSIVE'
undo_tablespace='UNDOTBS1'
LOG_ARCHIVE_DES_1='location=/home/oracle/project/disk10/archive'
LOG_ARCHIVE_FORMAT=PROJECT_%t_%s_%r.ARC
```

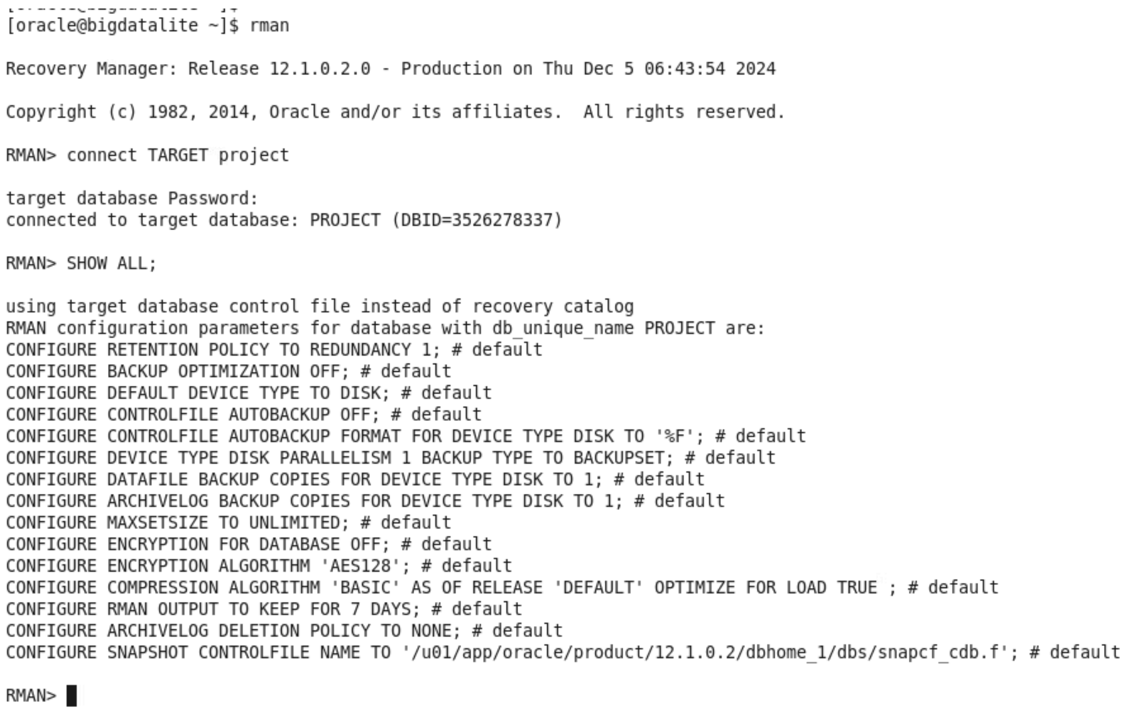
To prepare the database for recovery, I edited the database parameter file to include the destination of the log archive file and specified the format of the file.



```
SQL> startup pfile=/home/oracle/project/disk1/admin/pfile/initproject.ora MOUNT
ORACLE instance started.

Total System Global Area 1073741824 bytes
Fixed Size                  2932632 bytes
Variable Size             637534312 bytes
Database Buffers          427819008 bytes
Redo Buffers                5455872 bytes
Database mounted.
SQL> alter database archivelog;

Database altered.

SQL> alter database open;

Database altered.

SQL> archive log list;
Database log mode              Archive Mode
Automatic archival             Enabled
Archive destination            /home/oracle/project/disk10/archive
Oldest online log sequence     1252
Next log sequence to archive   1254
Current log sequence           1254
SQL> 
```

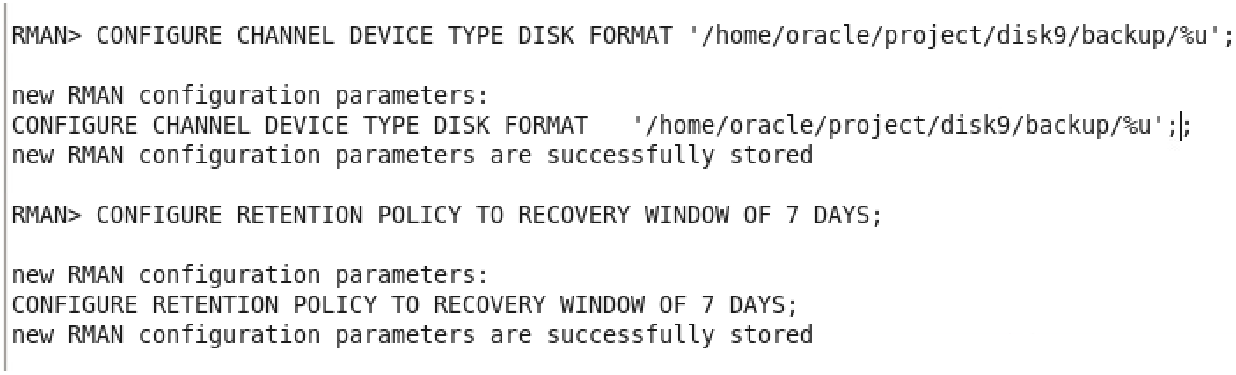
I then mounted the database without opening it and enabled archivelog mode so that the database could be backed up. I then opened the database up.

Document your experience with RMAN



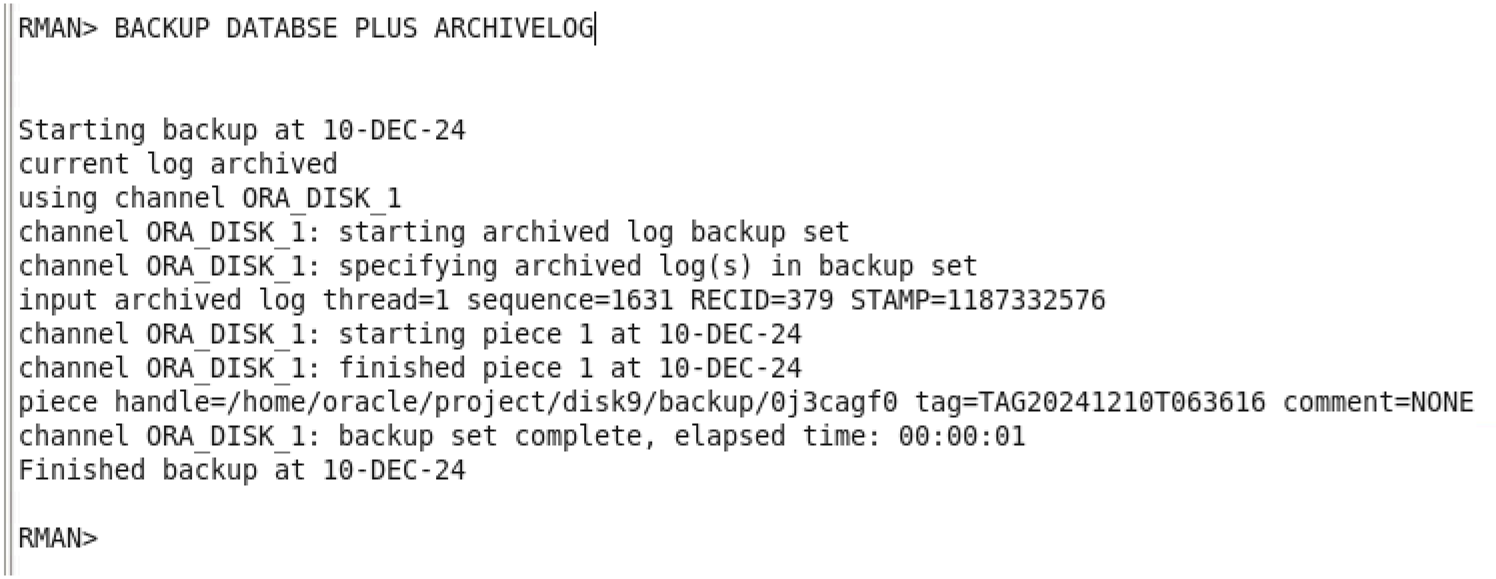
```
[oracle@bigdatalite ~]$ rman

Recovery Manager: Release 12.1.0.2.0 - Production on Thu Dec 5 06:43:54 2024

Copyright (c) 1982, 2014, Oracle and/or its affiliates.  All rights reserved.

RMAN> connect TARGET project

target database Password:
connected to target database: PROJECT (DBID=3526278337)

RMAN> SHOW ALL;

using target database control file instead of recovery catalog
RMAN configuration parameters for database with db_unique_name PROJECT are:
CONFIGURE RETENTION POLICY TO REDUNDANCY 1; # default
CONFIGURE BACKUP OPTIMIZATION OFF; # default
CONFIGURE DEFAULT DEVICE TYPE TO DISK; # default
CONFIGURE CONTROLFILE AUTOBACKUP OFF; # default
CONFIGURE CONTROLFILE AUTOBACKUP FORMAT FOR DEVICE TYPE DISK TO '%F'; # default
CONFIGURE DEVICE TYPE DISK PARALLELISM 1 BACKUP TYPE TO BACKUPSET; # default
CONFIGURE DATAFILE BACKUP COPIES FOR DEVICE TYPE DISK TO 1; # default
CONFIGURE ARCHIVELOG BACKUP COPIES FOR DEVICE TYPE DISK TO 1; # default
CONFIGURE MAXSETSIZE TO UNLIMITED; # default
CONFIGURE ENCRYPTION FOR DATABASE OFF; # default
CONFIGURE ENCRYPTION ALGORITHM 'AES128'; # default
CONFIGURE COMPRESSION ALGORITHM 'BASIC' AS OF RELEASE 'DEFAULT' OPTIMIZE FOR LOAD TRUE ; # default
CONFIGURE RMAN OUTPUT TO KEEP FOR 7 DAYS; # default
CONFIGURE ARCHIVELOG DELETION POLICY TO NONE; # default
CONFIGURE SNAPSHOT CONTROLFILE NAME TO '/u01/app/oracle/product/12.1.0.2/dbhome_1/dbs/snapcf_cdb.f'; # default

RMAN>
```

To use `RMAN` to backup my database, I first connected `RMAN` to my database using `connect TARGET project` in the RMAN console.



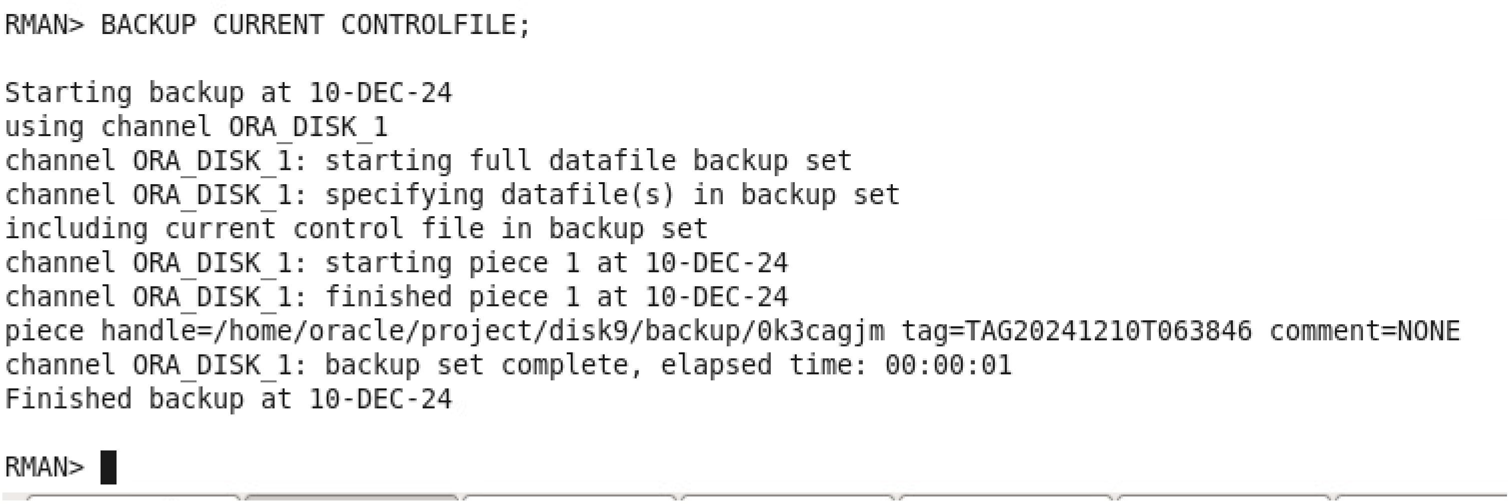
```
RMAN> CONFIGURE CHANNEL DEVICE TYPE DISK FORMAT '/home/oracle/project/disk9/backup/%u';

new RMAN configuration parameters:
CONFIGURE CHANNEL DEVICE TYPE DISK FORMAT   '/home/oracle/project/disk9/backup/%u';;
new RMAN configuration parameters are successfully stored

RMAN> CONFIGURE RETENTION POLICY TO RECOVERY WINDOW OF 7 DAYS;

new RMAN configuration parameters:
CONFIGURE RETENTION POLICY TO RECOVERY WINDOW OF 7 DAYS;
new RMAN configuration parameters are successfully stored
```

The next step was to edit the `RMAN` settings, configure where the backup would be saved, and configure how long the backup would be saved for in order to recover the database.



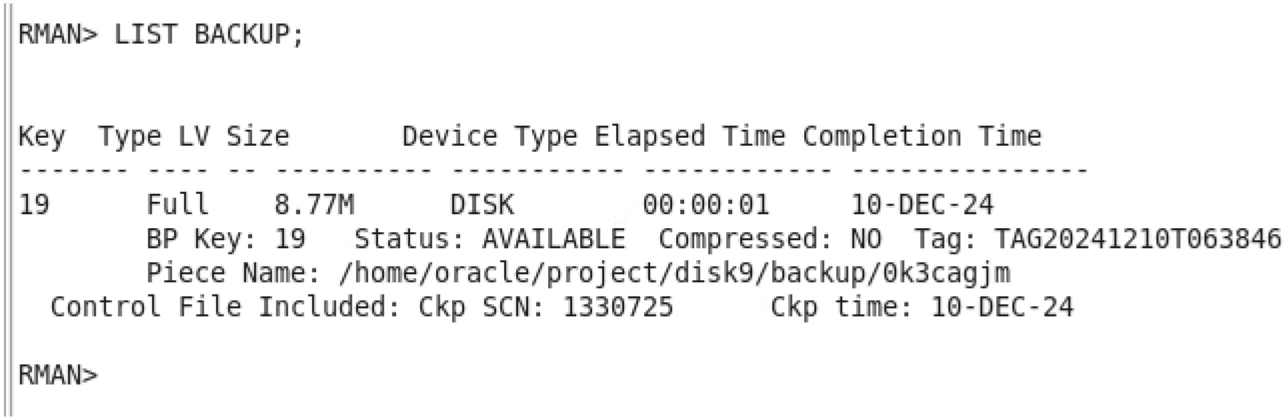
```
RMAN> BACKUP DATABSE PLUS ARCHIVELOG


Starting backup at 10-DEC-24
current log archived
using channel ORA_DISK_1
channel ORA_DISK_1: starting archived log backup set
channel ORA_DISK_1: specifying archived log(s) in backup set
input archived log thread=1 sequence=1631 RECID=379 STAMP=1187332576
channel ORA_DISK_1: starting piece 1 at 10-DEC-24
channel ORA_DISK_1: finished piece 1 at 10-DEC-24
piece handle=/home/oracle/project/disk9/backup/0j3cagf0 tag=TAG20241210T063616 comment=NONE
channel ORA_DISK_1: backup set complete, elapsed time: 00:00:01
Finished backup at 10-DEC-24

RMAN>
```

To backup the database, I ran the `BACKUP DATABASE PLUS ARCHIVELOG;` command in the RMAN console to back up the database and the archive log file.



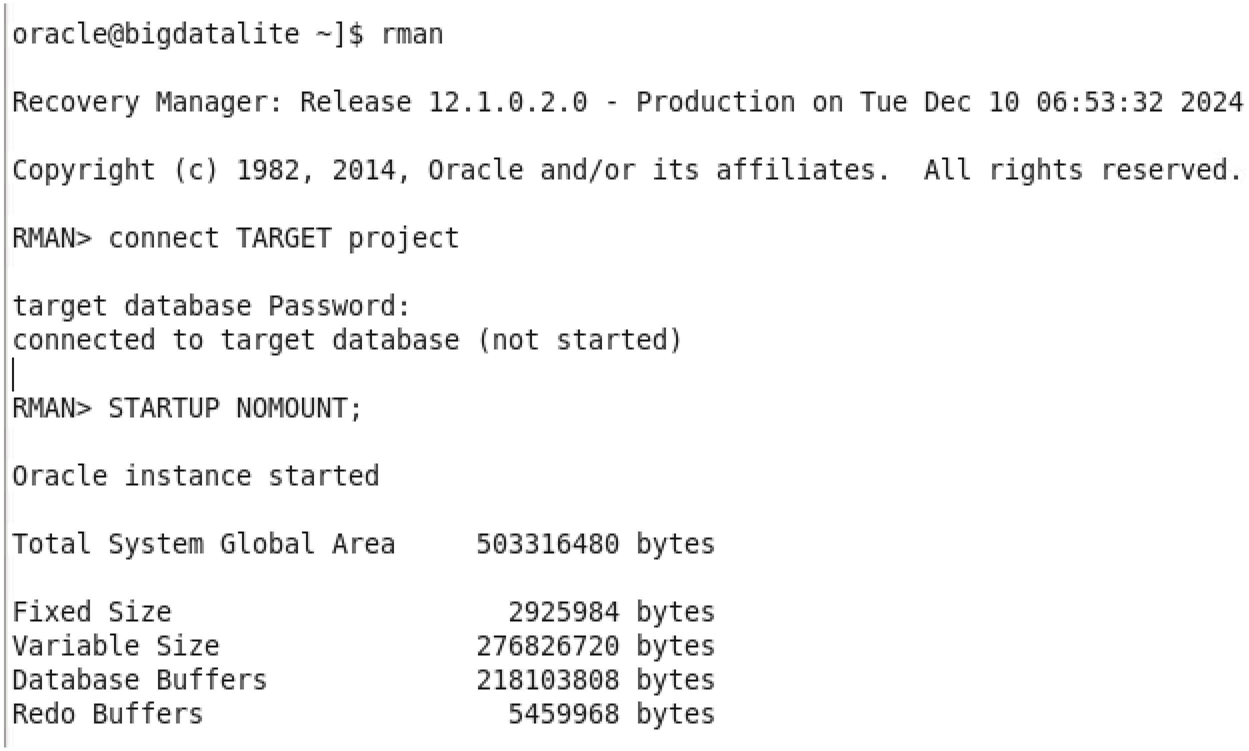
```
RMAN> BACKUP CURRENT CONTROLFILE;

Starting backup at 10-DEC-24
using channel ORA_DISK_1
channel ORA_DISK_1: starting full datafile backup set
channel ORA_DISK_1: specifying datafile(s) in backup set
including current control file in backup set
channel ORA_DISK_1: starting piece 1 at 10-DEC-24
channel ORA_DISK_1: finished piece 1 at 10-DEC-24
piece handle=/home/oracle/project/disk9/backup/0k3cagjm tag=TAG20241210T063846 comment=NONE
channel ORA_DISK_1: backup set complete, elapsed time: 00:00:01
Finished backup at 10-DEC-24

RMAN> 
```

I also backed up the current control file using the `BACKUP CURRENT CONTROLFILE;` in the RMAN console.



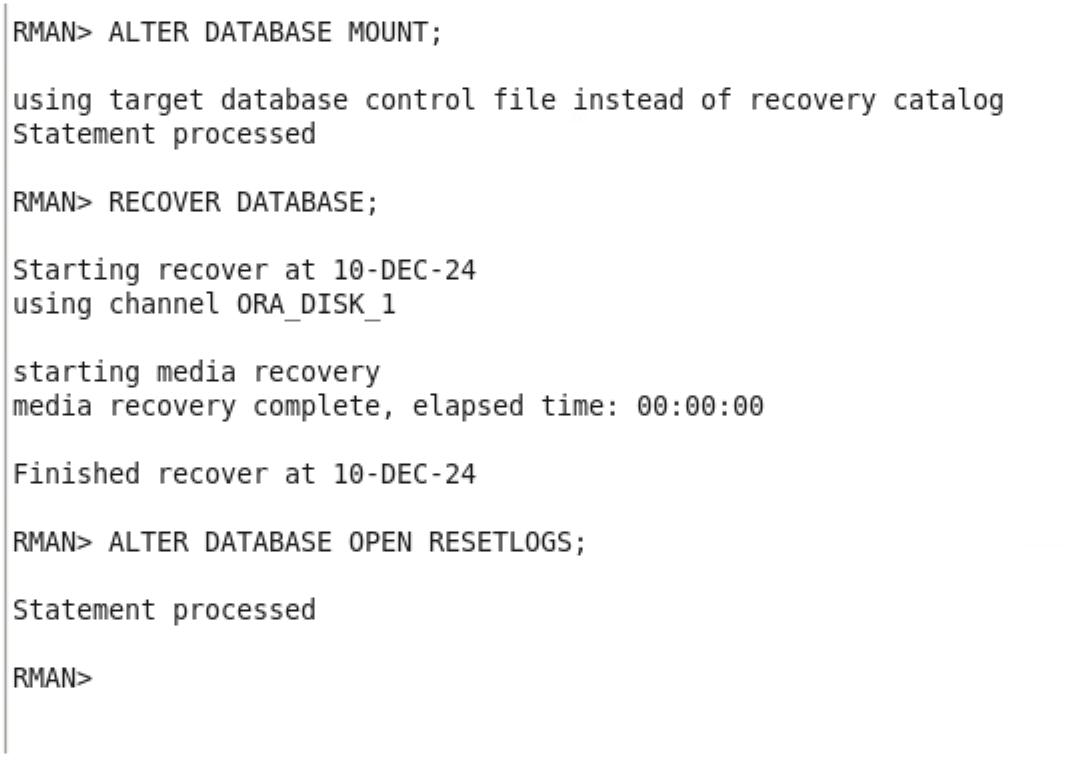
```
RMAN> LIST BACKUP;


Key  Type LV Size       Device Type Elapsed Time Completion Time
------- ---- -- ---------- ----------- ------------ ---------------
19      Full    8.77M      DISK        00:00:01     10-DEC-24
        BP Key: 19   Status: AVAILABLE  Compressed: NO  Tag: TAG20241210T063846
        Piece Name: /home/oracle/project/disk9/backup/0k3cagjm
  Control File Included: Ckp SCN: 1330725      Ckp time: 10-DEC-24

RMAN>
```

To verify that the backup was successfully created, I ran the `LIST BACKUP;` command which as seen above, there has been a successful backup.



```
oracle@bigdatalite ~]$ rman

Recovery Manager: Release 12.1.0.2.0 - Production on Tue Dec 10 06:53:32 2024

Copyright (c) 1982, 2014, Oracle and/or its affiliates.  All rights reserved.

RMAN> connect TARGET project

target database Password:
connected to target database (not started)

RMAN> STARTUP NOMOUNT;

Oracle instance started

Total System Global Area     503316480 bytes

Fixed Size                     2925984 bytes
Variable Size                276826720 bytes
Database Buffers             218103808 bytes
Redo Buffers                   5459968 bytes
```

To recover the database, I first shutdown the database and connected to it through RMAN.

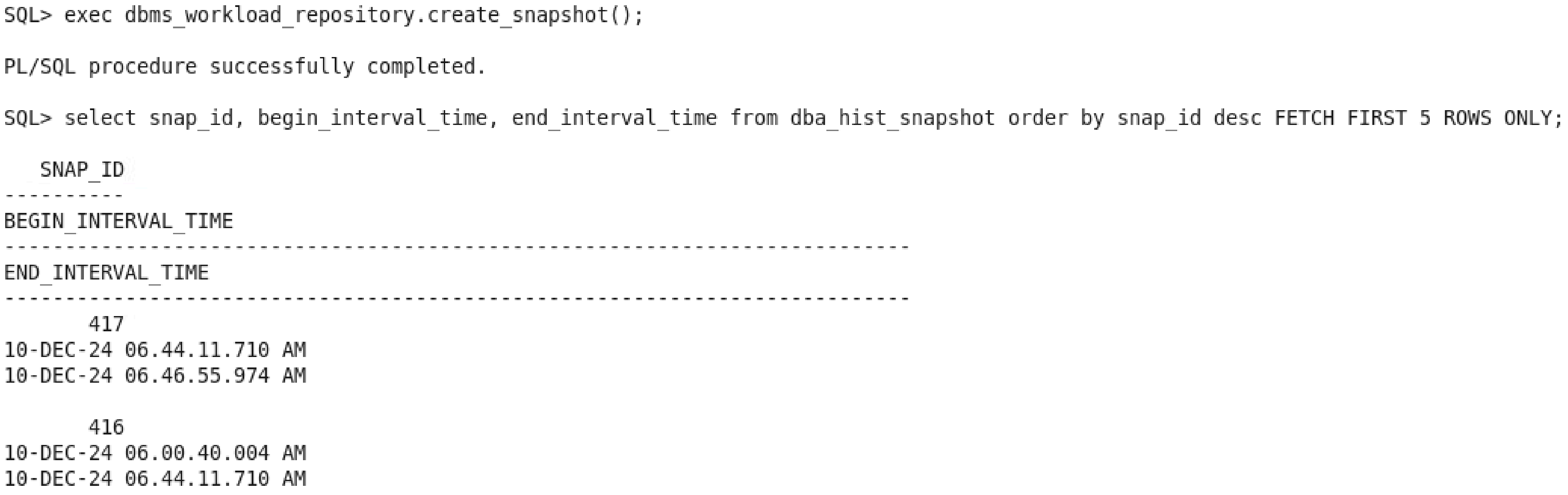


```
RMAN> ALTER DATABASE MOUNT;

using target database control file instead of recovery catalog
Statement processed

RMAN> RECOVER DATABASE;

Starting recover at 10-DEC-24
using channel ORA_DISK_1

starting media recovery
media recovery complete, elapsed time: 00:00:00

Finished recover at 10-DEC-24

RMAN> ALTER DATABASE OPEN RESETLOGS;

Statement processed

RMAN>
```

I then mounted the database and issued the `RECOVER DATABASE;` command which, as seen above, was successful. I then opened the database and archived and reset the redo logs.

Noah Zhou

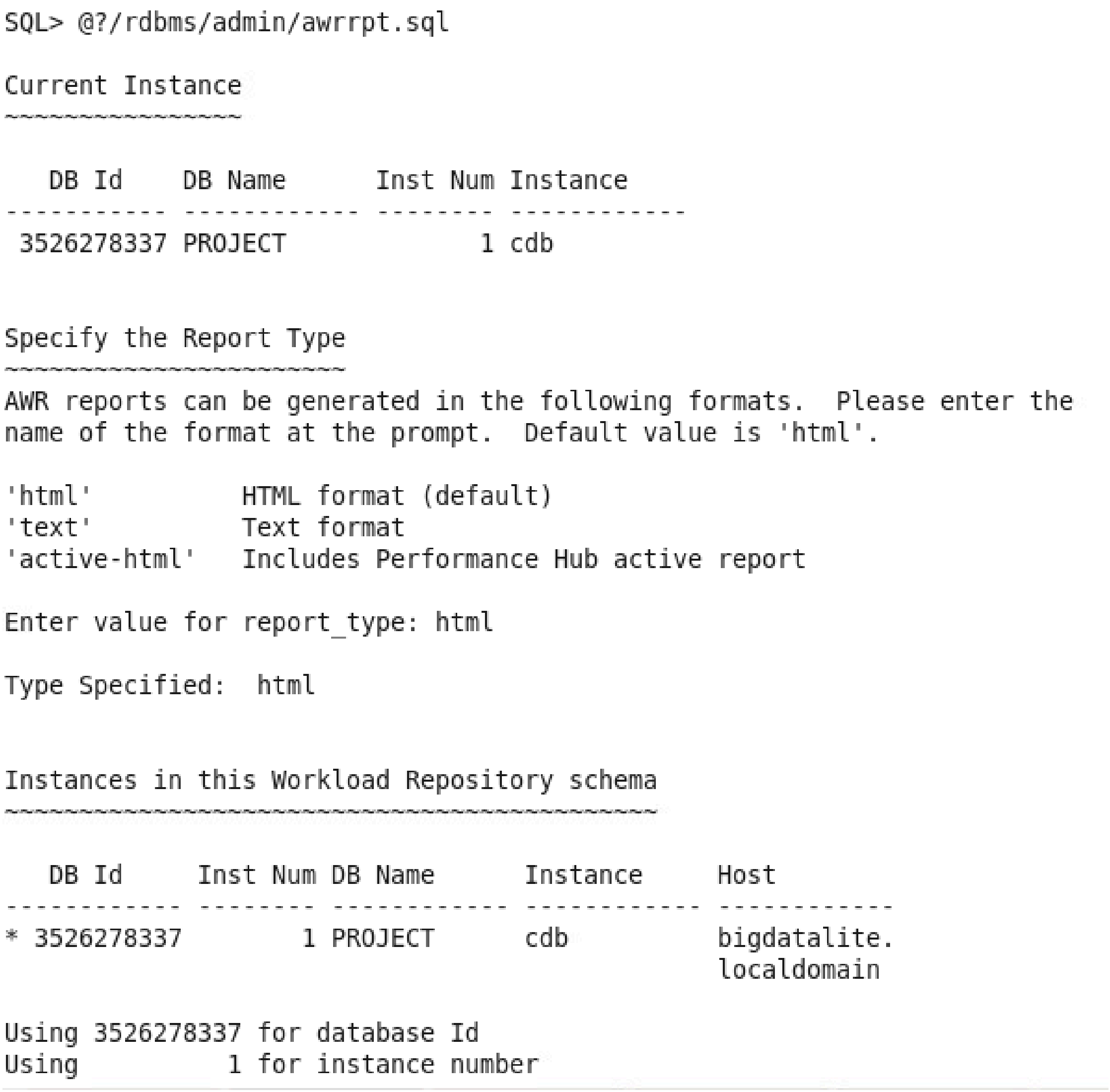
Benchmark query performance by capturing server data, tools such as AWR snapshot



```
SQL> exec dbms_workload_repository.create_snapshot();

PL/SQL procedure successfully completed.

SQL> select snap_id, begin_interval_time, end_interval_time from dba_hist_snapshot order by snap_id desc FETCH FIRST 5 ROWS ONLY;

   SNAP_ID
----------
BEGIN_INTERVAL_TIME
---------------------------------------------------------------------------
END_INTERVAL_TIME
---------------------------------------------------------------------------
       417
10-DEC-24 06.44.11.710 AM
10-DEC-24 06.46.55.974 AM

       416
10-DEC-24 06.00.40.004 AM
10-DEC-24 06.44.11.710 AM
```

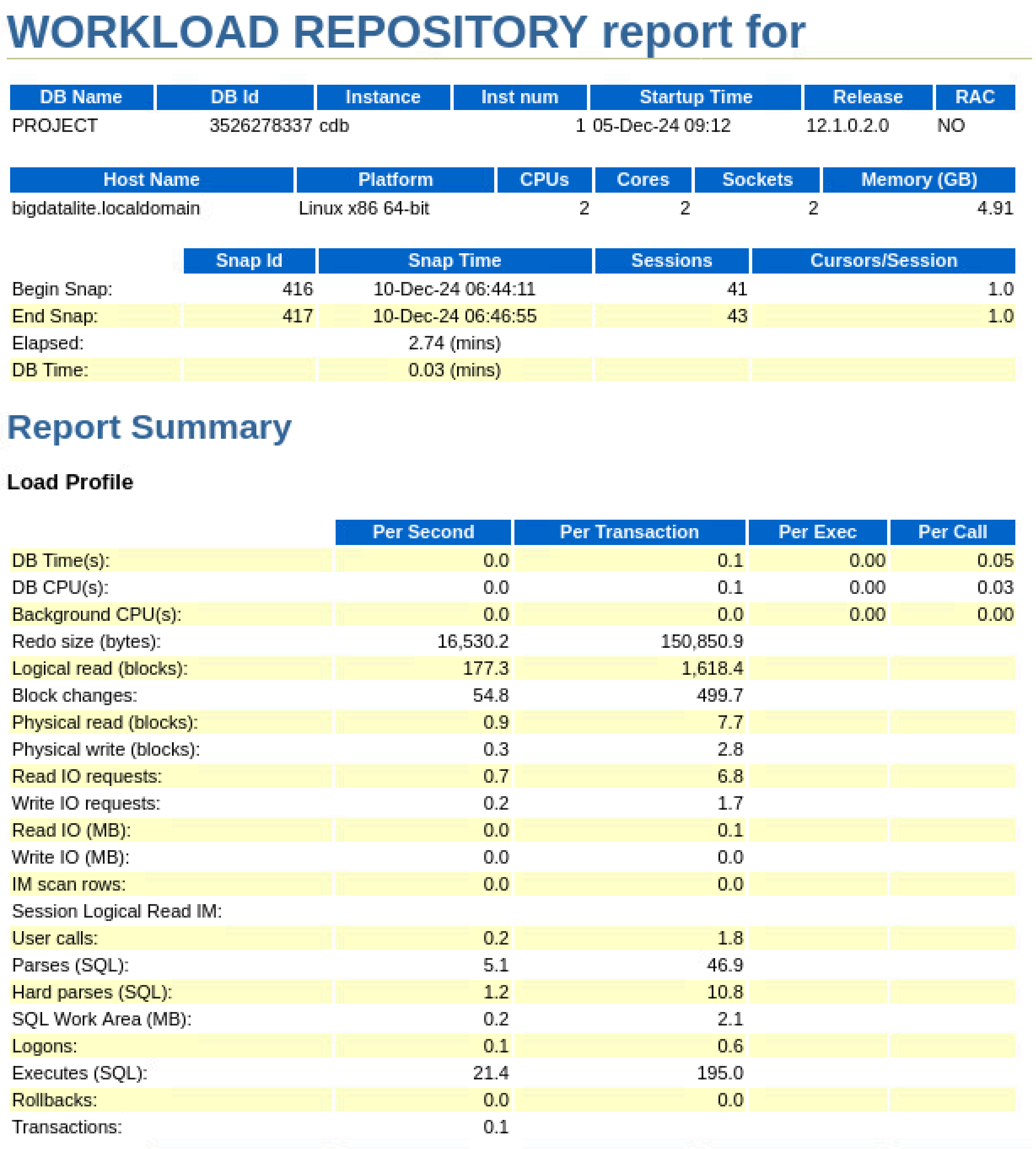
To create the snapshot, I ran the `exec dbms_workload_repository.create_snapshot();` command. I then queried the `dba_hist_snapshot` table to verify that I had successfully created a snapshot.

Noah Zhou



```
SQL> @?/rdbms/admin/awrrpt.sql

Current Instance
~~~~~~~~~~~~~~~~

   DB Id    DB Name      Inst Num Instance
----------- ------------ -------- ------------
 3526278337 PROJECT             1 cdb


Specify the Report Type
~~~~~~~~~~~~~~~~~~~~~~~
AWR reports can be generated in the following formats.  Please enter the
name of the format at the prompt.  Default value is 'html'.

'html'          HTML format (default)
'text'          Text format
'active-html'   Includes Performance Hub active report

Enter value for report_type: html

Type Specified:  html


Instances in this Workload Repository schema
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

   DB Id     Inst Num DB Name      Instance     Host
------------ -------- ------------ ------------ ------------
* 3526278337        1 PROJECT      cdb          bigdatalite.
                                                localdomain

Using 3526278337 for database Id
Using          1 for instance number
```

To access the AWR report, I ran the `@?/rdbms/admin/awrrpt.sql` command, and filled in the necessary information. For the format, I chose HTML as that would be the easiest to view.



WORKLOAD REPOSITORY report for

DB Name	DB Id	Instance	Inst num	Startup Time	Release	RAC
PROJECT	3526278337	cdb	1	05-Dec-24 09:12	12.1.0.2.0	NO

Host Name	Platform	CPUs	Cores	Sockets	Memory (GB)
bigdatalite.localdomain	Linux x86 64-bit	2	2	2	4.91

	Snap Id	Snap Time	Sessions	Cursors/Session
Begin Snap:	416	10-Dec-24 06:44:11	41	1.0
End Snap:	417	10-Dec-24 06:46:55	43	1.0
Elapsed:		2.74 (mins)		
DB Time:		0.03 (mins)		

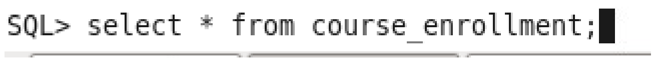
Report Summary

Load Profile

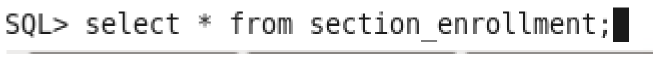
	Per Second	Per Transaction	Per Exec	Per Call
DB Time(s):	0.0	0.1	0.00	0.05
DB CPU(s):	0.0	0.1	0.00	0.03
Background CPU(s):	0.0	0.0	0.00	0.00
Redo size (bytes):	16,530.2	150,850.9		
Logical read (blocks):	177.3	1,618.4		
Block changes:	54.8	499.7		
Physical read (blocks):	0.9	7.7		
Physical write (blocks):	0.3	2.8		
Read IO requests:	0.7	6.8		
Write IO requests:	0.2	1.7		
Read IO (MB):	0.0	0.1		
Write IO (MB):	0.0	0.0		
IM scan rows:	0.0	0.0		
Session Logical Read IM:				
User calls:	0.2	1.8		
Parses (SQL):	5.1	46.9		
Hard parses (SQL):	1.2	10.8		
SQL Work Area (MB):	0.2	2.1		
Logons:	0.1	0.6		
Executes (SQL):	21.4	195.0		
Rollbacks:	0.0	0.0		
Transactions:	0.1			

Noah Zhou

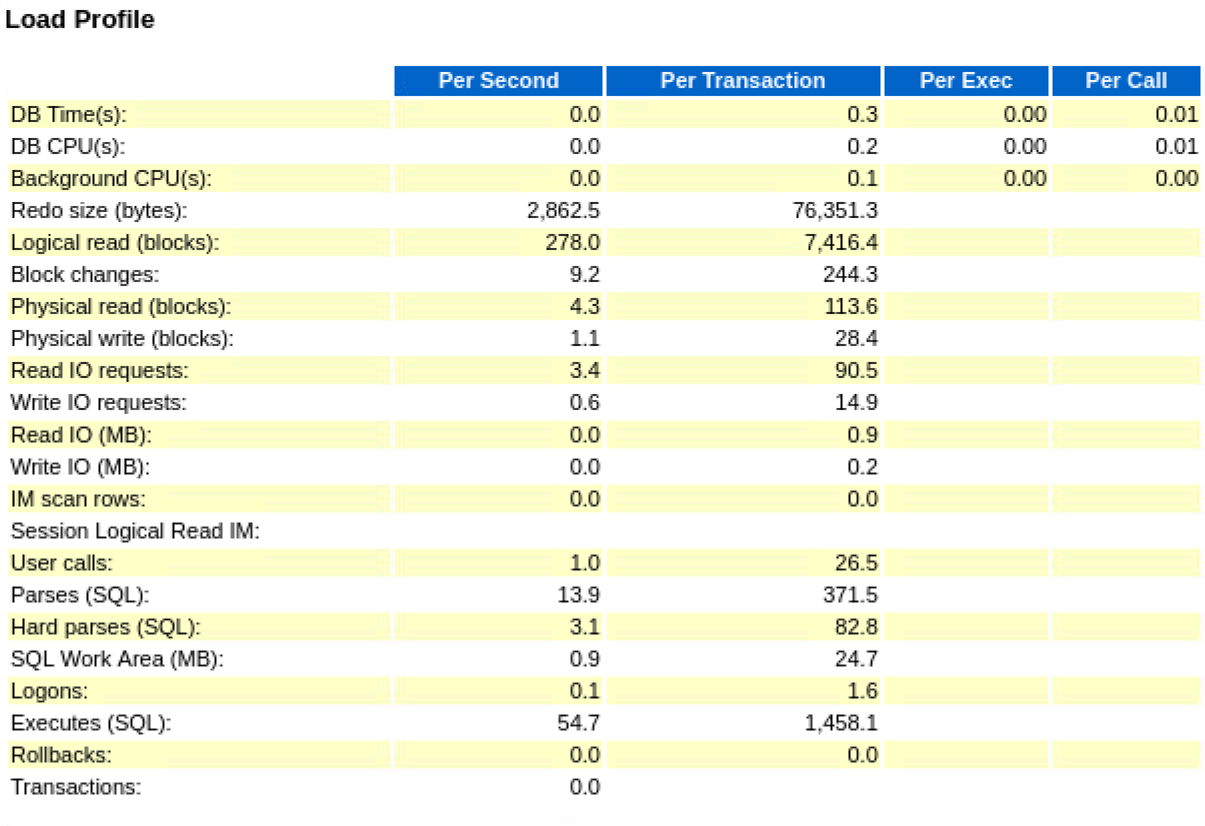
This figure is the HTML AWR report. It shows the performance of the database before running any queries.



```
SQL> select * from course_enrollment;
```



```
SQL> select * from section_enrollment;
```



Load Profile

	Per Second	Per Transaction	Per Exec	Per Call
DB Time(s):	0.0	0.3	0.00	0.01
DB CPU(s):	0.0	0.2	0.00	0.01
Background CPU(s):	0.0	0.1	0.00	0.00
Redo size (bytes):	2,862.5	76,351.3		
Logical read (blocks):	278.0	7,416.4		
Block changes:	9.2	244.3		
Physical read (blocks):	4.3	113.6		
Physical write (blocks):	1.1	28.4		
Read IO requests:	3.4	90.5		
Write IO requests:	0.6	14.9		
Read IO (MB):	0.0	0.9		
Write IO (MB):	0.0	0.2		
IM scan rows:	0.0	0.0		
Session Logical Read IM:				
User calls:	1.0	26.5		
Parses (SQL):	13.9	371.5		
Hard parses (SQL):	3.1	82.8		
SQL Work Area (MB):	0.9	24.7		
Logons:	0.1	1.6		
Executes (SQL):	54.7	1,458.1		
Rollbacks:	0.0	0.0		
Transactions:	0.0			

Noah Zhou

I ran two queries that printed out a large number of rows and then retook the snapshot and recreated the report, again in the HTML format. We can see that after running the queries, there was a significant increase in Redo Size, Logical Read, User Calls, Parses (SQL), Executes (SQL), and more.