A Blueprint for Database Design and Development

HANDS ON PRACTICE WORKBOOK

Ekokobe, Imgard
KUDRANT ACADEMY

Table of Contents

Read Me First

Introducing you to the fundamental principles of Database Design and Development.

Now that you have learned the strategies and principles of database design and development, you will use these to design a small database.

These fundamental concepts are necessary for the design, use, and implementation of relational database systems such as the Oracle database management system (DBMS).

This lab will help you develop skills for database design, development, creation of the physical database on disks and query formulation.

You will use a data modelling tool called ER Assistant to design a small database (about 5 tables) by developing an Entity Relationship Diagram (ERD).

ERD is a graphical way of presenting the database that's going to be developed. Once you design the ERD, you're going to translate that into DDL – data definition language, so you'll be going from the graphical representation to actually defining tables and columns.

Graphical representations are important because it is more cost effective and efficient to modify a database design on an ERD than it is to change database objects after they have been created on disk.

You will use this DDL statements to create your database objects by running them against your database.

Once you have created your database objects, you can load some data into the tables and you can create views (also called virtual tables).

You will be able to retrieve the data you loaded into your database to answer a question using simple SQL select statements.

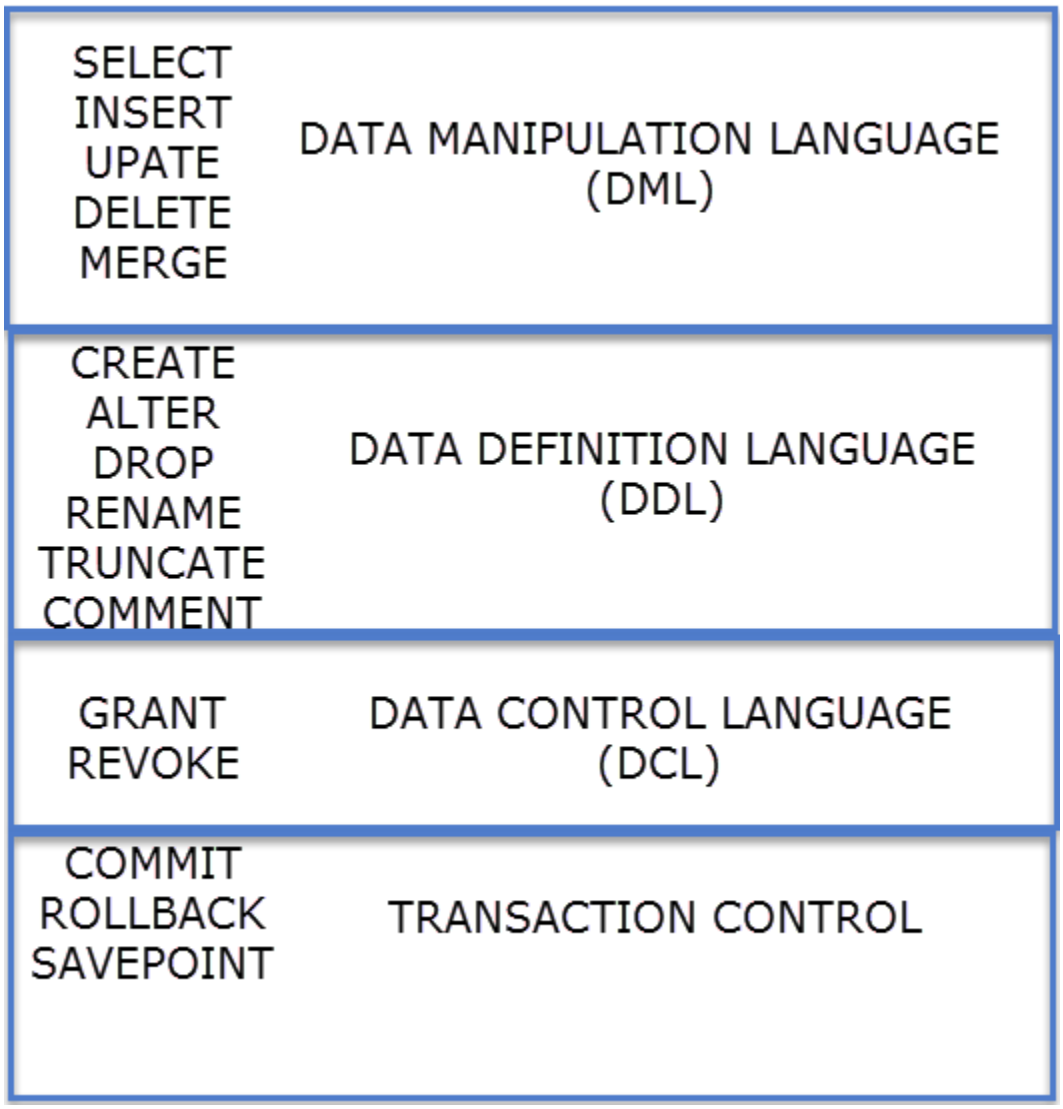
This is the fundamental level of database design and development.

The exercises and solutions in this Lab simulate real-life database design and development goals.

Every database has a special interface between end-users and the database engine. In Oracle, this special interface is called SQL*Plus. The basic function of the interface is to allow users to execute commands to the database engine. You will connect to your database, create a user account and use run the DDL statements to create your database objects from this account. Once you have created your database objects, you will use DML statement to load some data into the tables and you will be able to retrieve the data by using simple SQL select statements.

Keep in mind!

The Database Management System (DBMS) you will use for this project will be Oracle.


SELECT
INSERT
UPATE
DELETE
MERGE
DATA MANIPULATION LANGUAGE (DML)
CREATE
ALTER
DROP
RENAME
TRUNCATE
COMMENT
DATA DEFINITION LANGUAGE (DDL)
GRANT
REVOKE
DATA CONTROL LANGUAGE (DCL)
COMMIT
ROLLBACK
SAVEPOINT
TRANSACTION CONTROL

Objectives

- Identify the purpose of your database
- Identify the entities and attributes of your database tables (five tables).
- Define the constraints for your tables.
- Create an Entity Relationship Diagram (ERD) for a small database of five tables.
- Translate the ERD into tables.
- Use SQL*Plus to connect to an oracle database.
- Create physical objects in a database using a script.
- Create a user account.
- Load data into tables using a script.
- Retrieve data from tables.
- Capture output to a log file.

Tools Used

Databases are carefully planned and designed after gathering and analyzing user requirements from stakeholders.

After the requirements, have been gathered and defined, a model will be created for the database.

This model will be used to use to create the Entity Relationship Diagrams (ERD) and these ERDs will then be translated into Data Definition Languages (DDL).

The DDL script will be used to create your database tables and the DML script will be used to load some data into your tables.

There are some tools to help you create the necessary database tables.

These tools include:

- An editor (notepad in your case).
- SQL*Plus, and capabilities to print your source code and capture the output.
- ER Assistant tool.

You will learn how to use these tools in this workbook.

Running a Script in SQL*plus

You have learned how to execute DDL statements to create the tables dynamically within SQL*Plus.

On the job, these database objects are not usually created dynamically. You will be provided with scripts which you will review, modify and finalize with the database developers.

What are the benefits of Using Scripts versus entering the DDL commands on the command line create your database tables one at a time?

- The script will serve as documentation on the characteristics of the objects.
- Any errors in the syntax can be fixed and the script re-run until it is clean or free of errors.
- The script can be re-used or replicated across the database environments without any issues since they would already have been tested and re-tested.

How to Create Tables Using a Script

1. Enter the SQL statements in a text file.
2. Save the file with the *.sql file extension.
3. Save this file to a folder on your machine. You may name the folder dba_scripts. This is where all your sql scripts will be stored.
4. Open up SQL*Plus
5. Run the script by typing "@filename" at the sql> prompt. For example if my DDL script is located under the C:\Users\Imgarde\Documents\dbascripts folder. I will type the following at the SQL prompt

   sql> @C:\Users\Imgarde\Documents\dbascripts\unidb.sql

   Note: There should be **NO SPACES** between the **@** sign and the filename or path to the file when running the script.

Two Ways to Capture the Output of a DDL Script

1. Screen Capture

The simplest way to capture the output on the screen is to highlight it with the mouse. Then copy, paste it to a text file (notepad) and save it.

2. Printing the output to a file using the Spool Command

This command is used to record all of the screen displays that would normally run and put them in a file. It is useful to capture the screen output or displays to verify that everything runs correctly.

Two ways to run the SPOOL command to log the output of your DDL (create table script) and DML (script to insert the data in your tables) scripts:

a. **You can type the SPOOL command directly from the command line.**

☺ At the SQL prompt, you will type the following command which will start printing the output to a file called imgard_createunidb.log in the directory you specified.

```
SQL> spool C:\Users\Imgarde\Documents\dba_scripts\imgard_createunidb.log
```

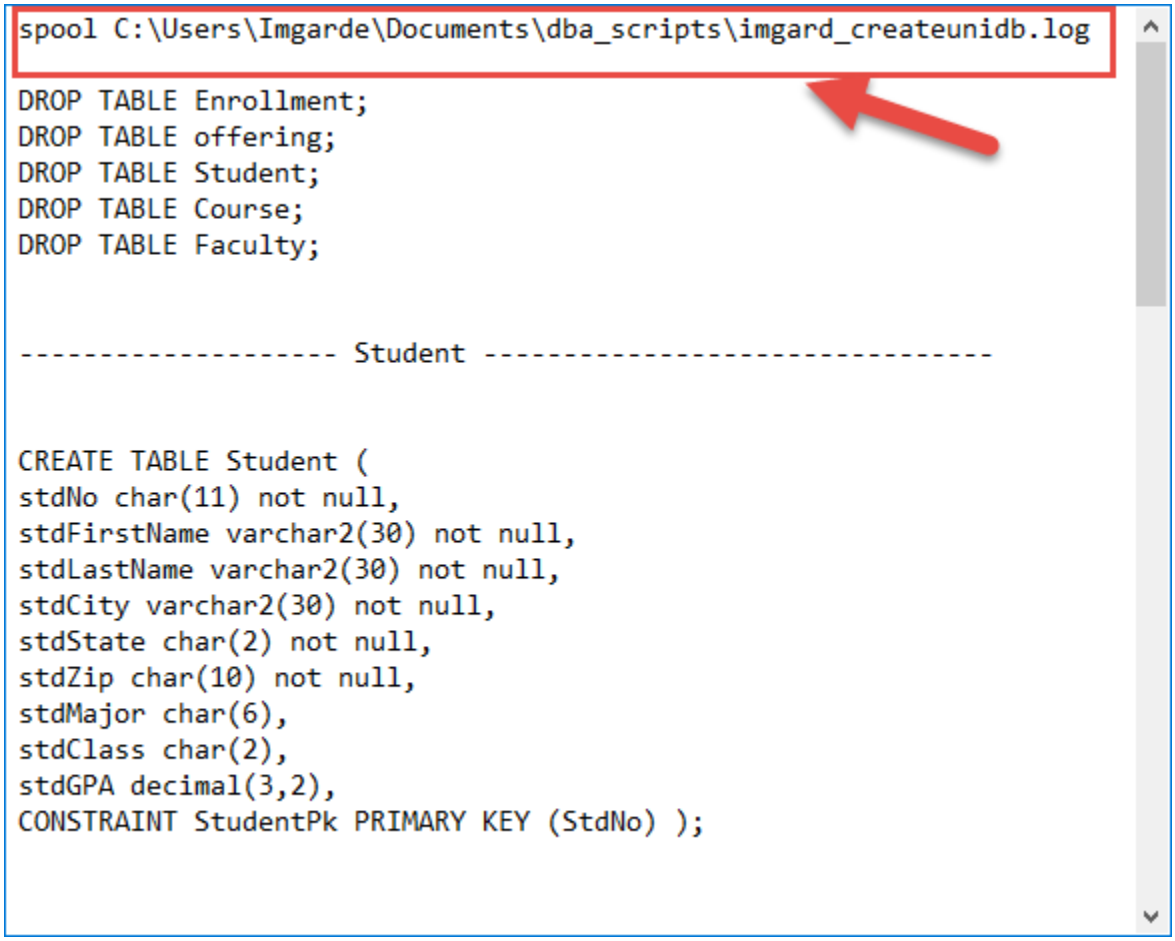
☺ At the next SQL prompt, you will type the following command to run the DDL script to create your 5 tables.

```
SQL> @C:\Users\Imgarde\Documents\dba_scripts\TeamName_ddl.sql
```

☺ At the next SQL prompt, you will type the spool off command to stop print to the file.

```
SQL> spool off
```

b. **You can type the SPOOL command within the DDL script (preferred method for this assignment).**



```
spool C:\Users\Imgarde\Documents\dba_scripts\imgard_createunidb.log

DROP TABLE Enrollment;
DROP TABLE offering;
DROP TABLE Student;
DROP TABLE Course;
DROP TABLE Faculty;


-------------------- Student --------------------------------


CREATE TABLE Student (
stdNo char(11) not null,
stdFirstName varchar2(30) not null,
stdLastName varchar2(30) not null,
stdCity varchar2(30) not null,
stdState char(2) not null,
stdZip char(10) not null,
stdMajor char(6),
stdClass char(2),
stdGPA decimal(3,2),
CONSTRAINT StudentPk PRIMARY KEY (StdNo) );
```

Task 1 – Statement of Work

The Statement of Work (SOW) is a document which describes the scope of work required to complete a project. It provides the background and requirements of the project, the stakeholder's roles and responsibilities, milestone of the projects, the timeframe for completion of project and deliverables.

Lab Assignment for Task 1

Write a paragraph describing the database you will design and implement. Describe the business solution your database will solve and how it will be used by the end users. Include the following information -

- Purpose of the database.
- The hardware your database will run on.
- The database management system you will use (include the version).
- The type of Data Definition Language (DDL) and Data Manipulation Language (DML) you plan to use.
- The tools you will need.
- How you will access your database.

Sample Solution!

1. The database will serve as a prototype for a university management system. It will store data about students, faculty, courses, course offerings, and enrollments. It will track student registrations, faculty assignment to course offerings, the scheduling of course offerings and student grades.
2. My database platform will be Windows 10.
3. The DBMS will be Oracle 11gR2 (11.2.0.1).
4. The DDL and DML will be SQL Create and Insert and Select statements.
5. I will use Notepad for creating and editing SQL statements and the ER Assistant tool to design the ERD.
6. I will access oracle using SQL*plus running from my windows machine to connect to my oracle database locally.

Deliverable 1- Statement of Work (SOW)

★ **One page double-spaced word document including the following information.**

- Purpose of the database.
- The hardware your database will run on.
- The database management system you will use (include the version).
- The type of Data Definition Language (DDL) and Data Manipulation Language (DML) you plan to use.
- The tools you will need.
- How you will access your database.

Document Name: TeamName_SOW

Task 2 – Create ERD

The Entity-Relationship Diagram (ERD) is a communication tool. This is an opportunity to communicate your design for technical review. In industry, this preliminary design is often subject to peer review through a process known as a *walk-through*. When you develop the design, think of it as preparation for a presentation to your customer or for a formal walk-through with the customer, analysts, system architects, and developers who will implement your design.

Lab Assignment for Task 2

1. Develop an ERD using the ER Assistant tool you downloaded. You will create entity diagram types for your 5 tables which will depict the entities, attributes, primary keys, foreign keys and the relationships between the tables. Each table must be normalized to the third normal form.
2. The first step is to identify the information the customer will need to store in the database.

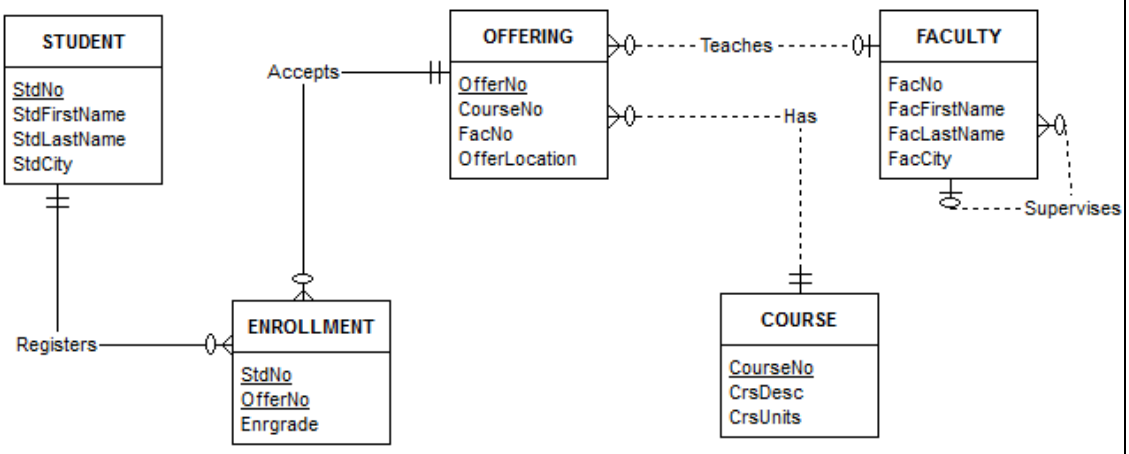
★DBA Tip!
★ Be sure to properly define your primary and foreign keys in your ERD. ★ Review your design to fix all the kinks. Logical designs can be easily modified compared to physical design.

Checklist

- First, aggregate the information you want to store. This means that you put information that seems to go together into one entity. Looking at the data, five logical groupings emerge: student, course, offering, faculty and enrollment information. The logical groupings become the entities and are shown as rectangles on the ERD.
- Establish the relationships between the entities. The relationships can be one-to-one, one-to-many, or many-to-many.
- Define the attributes for all 5 entities.
- Define the primary keys for each entity (an attribute or group of attributes which will make each row unique. In the sample ERD for the university database; for the Student table, StdNo is unique, CourseNo is unique for the Course table, OfferNo is unique for the Offering table, FacNo is unique for the Faculty table and the combination of StdNo and OfferNo are unique for the Enrollment table.
- Define the foreign keys for each entity.
- Assign non-key attributes to all 5 entities for example StdFirstName and StdLastName for the Student table.
- Normalize the tables. Apply the normalization rules to the tables in the ERD. Make any changes needed until the design is normalized to third normal form. Notice that the "crows foot" is used to show the "many" side of the relationship while a plain line shows the "one" sided relationship.

Sample Solution!

Here is a sample ERD for the University Database we studied in class



Deliverable 2 - ERD

★ Submit Entity Relationship diagram for your normalized ERD in a word document with a description of the entities and the relationships between them.

Document Name: TeamName_ERD

Task 3 - Convert ERD to Tables (DDL script)

Now that you have designed a normalized Entity Relationship Diagram (ERD), your next task is to translate the logical design into physical objects that will reside on the physical disks.

To do this, you will write SQL DDL (Data Definition Language) statements to create your database objects. This process is frequently called *physical design*.

When the tables are created, each attribute on the ERD is also defined as a column for the tables. The data type of each column must be specified. You will use the CHAR, VARCHAR2, NUMBER, and DATE data types.

- If a column will contain alphanumeric data of a specific length, CHAR should be used.
- If the column will hold data that will be alphanumeric of varying length, VARCHAR2 should be used and the length of the longest value should be included.
- The NUMBER data type should be used if the column will hold only numbers and if mathematical calculations will be done on the column.
- DATE should be used as the data type if the column will hold dates or timestamps.

Lab Assignment for Task 3

1. Set up a DDL script to create your 5 tables for your database. Your DDL script will include the following:
   a. 5 CREATE TABLE statements to create your five tables.
   b. Include drop and create statements in proper order.
   c. Constraints defined for your tables (primary, foreign, unique, null, check).
   d. Define data types for each column of your tables.
   e. A view on a single table.
   f. A equijoin view
   g. Indexes as needed.
   h. A Query of the catalog on your database and be sure the five tables are shown.
2. Create a new user account in the database.
3. Execute the script in SQL*Plus against this new account.
4. Debug the script to clear any errors.

Checklist

- Set up your DDL script - Open up notepad to write your script to create your database objects. It is useful to capture the screen output or displays to verify that everything runs correctly. Script Name: TeamName_ddl.sql
- Create a user account and give privileges to allow user to connect to the database.

```
SQL> create user iekokobe identified by oracle123
default tablespace users
temporary tablespace temp;
SQL> grant connect, resource to iekokobe;
```

Note: Use your TeamName to create user account (We will learn about creating user accounts later). The password I have used is oracle123. Define a password for your user account.

- Add the spool command with the name of the spool file as the first line in your DDL script.
- Write 5 drop statements to drop your five tables. Since you may have to run the script more than once, these DROP commands are used to ensure that you start with a clean database. Notice that you have to drop the many-sides of the relationships first. When you drop the tables, the indexes will be dropped as well. The views on the tables will not be dropped, however, so you will create drop commands specifically for them.
- Write 5 create table statements to convert your 5 entities to 5 physical tables in the database.
- The parent tables will be created first followed by the child tables. For example from the university database ERD, the Parent tables are Student, Course and Faculty.
- The child tables are Offering and Enrolment because the offering table is related to the student table and the Enrollment table to the Course.
- Add SQL statements to create the indexes on the foreign keys. Notice that these indexes are not unique. Oracle created unique indexes on the primary keys so it was not necessary to do that explicitly.
- Add SQL statements to create two views. The views do not take up space because they will not contain data. That is, when they are executed, data will be shown but will not reside permanently in a table on disk.
- Add an SQL statement to check the catalog to be sure that the objects were created successfully.
- Add comments to the DDL script to explain the various objects. Comments are important because they help other designers working on the script

understand the purpose of it. The comments must be enclosed with a forward slash, an asterisk, the comment, an asterisk, and another forward slash as shown in the sample DDL script below.

- ❖ Debug the script

★DBA Tip!

1. When dropping tables using declarative referential integrity, drop the many-sided tables (child tables) first.

2. When creating tables using declarative referential integrity, the order of the tables is important. The "one-" sided tables (parent tables) have to be created before the "many-" sided tables (child tables). Otherwise, when the many-sided table references the one-sided, no target object will exist, generating an error.

3. When a table is created, space is allocated on disk for that object.

4. The structure of this script is generally standard in industry for new database development efforts.

Debug DDL Script

The DDL script may not run correctly the first time. The spooled file can be used to debug the script. Debugging a script simply means to find and fix all the errors generated in the log file. This is the reason why it is important to capture the output displayed in a log file. To debug your script:

1. Open the log file (generated from the spool tool) using a text editor (notepad).
2. Look for any errors in the file, oracle displays an error code which begins with ORA and a set of 5 digits for example (ORA-00942). Start with the first error in the spool file output and work it out until it clears. Clearing means you fix the error and rerun the script and check that it is clean or free of errors.

Guidelines for debugging scripts

1. Start at the top.

Always start at the top of the spooled log file and look at the first error. When the DDL script is run the first time, the following errors will be seen because the drop statements are at the beginning of the script:

For example:

```
DROP TABLE Student
           *
ERROR at line 1:
ORA-00942: table or view does not exist
```

```
SQL> @C:\Users\Imgarde\Documents\dba_scripts\unidb.sql
DROP TABLE Enrollment
           *
ERROR at line 1:
ORA-00942: table or view does not exist


DROP TABLE offering
           *
ERROR at line 1:
ORA-00942: table or view does not exist


DROP TABLE Student
           *
ERROR at line 1:
ORA-00942: table or view does not exist


DROP TABLE Course
           *
ERROR at line 1:
ORA-00942: table or view does not exist
```

This error is expected and can be ignored because you are trying to drop objects that do not yet exist in the database when you initially run your DDL script.

2. Check the line above the error

Look out for a similar error as shown below in your spool file. This error looks like there is something wrong with the primary key constraint. Notice that there is an asterisk below the part of the code that seems to be in error.

```
CONSTRAINT StudentPk PRIMARY KEY (StdNo) )
                                 *
ERROR at line 11:
ORA-00907: missing right parenthesis
```

```
DROP TABLE Faculty
           *
ERROR at line 1:
ORA-00942: table or view does not exist


CONSTRAINT StudentPk PRIMARY KEY (StdNo) )
                                 *
ERROR at line 11:
ORA-00907: missing right parenthesis


Table created.
```

Let's review the DDL script for the student table that generated the error:

```
CREATE TABLE Student (
stdNo char(11) not null,
stdFirstName varchar2(30) not null,
stdLastName varchar2(30) not null,
stdCity varchar2(30) not null,
stdState char(2) not null,
stdZip char(10) not null,
stdMajor char(6),
stdClass char(2),
stdGPA decimal(3,2) <--------error here
CONSTRAINT StudentPk PRIMARY KEY (StdNo) );
```

The problem occurred on the line above the constraint statement. Notice that there is a missing comma after the decimal (3, 2) for the StdGPA column.

Now let's review the modified DDL script as shown below:

```
CREATE TABLE Student (
stdNo char(11) not null,
stdFirstName varchar2(30) not null,
stdLastName varchar2(30) not null,
stdCity varchar2(30) not null,
stdState char(2) not null,
stdZip char(10) not null,
stdMajor char(6),
stdClass char(2),
stdGPA decimal(3,2),
CONSTRAINT StudentPk PRIMARY KEY (StdNo) );
```

Often, the error is generated by the line before the one shown in the error output. So, when debugging DDL scripts, it is often helps to look at the line above the one that is shown by the error message.

3. Check for unique names.

You could also see the following error in the spooled log file.

```
CONSTRAINT CourseFK FOREIGN KEY (CourseNo)
REFERENCES Course
           *
ERROR at line 35:
ORA-02264: name already used by an existing constraint
```

You will see this error if you define more than one foreign key constraints in your DDL script with identical names

You resolve the error by redefining a name for your foreign key constraint that is unique in the database.

★DBA Tip!

★ All objects such as tables, indexes and constraints must have unique names within the database.

Sample Solution!

Sample DDL

```
SPOOL TeamName_ddl.log
/*-------------------------------------------------------------------------*/
/*    DROP OBJECTS                                                         */
/*-------------------------------------------------------------------------*/
DROP TABLE Enrollment;
DROP TABLE offering;
DROP TABLE Student;
DROP TABLE Course;
DROP TABLE Faculty;
DROP VIEW STUDENT_VIEW;
DROP VIEW TWO_TABLES;
/*-------------------------------------------------------------------------*/
/*    CREATE TABLES                                                        */
/*-------------------------------------------------------------------------*/
CREATE TABLE Student
       (stdNo char(11)     not null,
        stdFirstName       varchar2(30) not null,
        stdLastName        varchar2(30) not null,
        stdCity            varchar2(30) not null,
        stdState           char(2) not null,
        stdZip             char(10) not null,
        stdMajor           char(6),
        stdClass           char(2),
        stdGPA             decimal(3,2),
        CONSTRAINT Pk_STUDENT PRIMARY KEY (StdNo)
        );


CREATE TABLE Course
       (CourseNo          char(6) not null,
        crsDesc           varchar2(50) not null,
        CrsUnits          integer,
        CONSTRAINT PK_COURSE PRIMARY KEY (CourseNo)
        );

CREATE TABLE Faculty
       (FacNo             char(11) not null,
        FacFirstName      varchar2(30) not null,
        FacLastName       varchar2(30) not null,
        FacCity           varchar2(30) not null,
        FacState          char(2) not null,
        FacZipCode        char(10) not null,
        FacRank           char(4),
        FacHireDate       date,
        FacSalary         decimal(10,2),
        FacSupervisor     char(11),
        FacDept           char(6),
        CONSTRAINT    PK_FACULTY PRIMARY KEY (FacNo),
```

```
       CONSTRAINT FK_SUPERVISOR FOREIGN KEY (FacSupervisor)
       REFERENCES Faculty
       );

CREATE TABLE Offering
       (OfferNo          INTEGER not null,
        CourseNo         char(6) not null,
        OffTerm          char(6) not null,
        OffYear          INTEGER not null,
        OffLocation      varchar2(30),
        OffTime          varchar2(10),
        FacNo            char(11),
        OffDays          char(4),
        CONSTRAINT PK_OFFERING PRIMARY KEY (OfferNo),
        CONSTRAINT FK_COURSE FOREIGN KEY (CourseNo) REFERENCES Course,
        CONSTRAINT FK_FACULTY FOREIGN KEY (FacNo) REFERENCES Faculty
       );

CREATE TABLE Enrollment
       (OfferNo          INTEGER not null,
        StdNo            char(11) not null,
        EnrGrade   decimal(3,2),
        CONSTRAINT PK_ENROLLMENT PRIMARY KEY (OfferNo, StdNo),
        CONSTRAINT FK_OFFERING FOREIGN KEY (OfferNo) REFERENCES Offering
        ON DELETE CASCADE,
        CONSTRAINT  FK_STUDENT  FOREIGN  KEY  (StdNo)  REFERENCES  Student  ON
        DELETE CASCADE
        );

/*--------------------------------------------------------------------------*/
/*    CREATE INDEXES                                                        */
/*--------------------------------------------------------------------------*/
CREATE INDEX FK_OFFERING
  on ENROLLMENT (OfferNo)
      ;
CREATE INDEX FK_STUDENT
  on ENROLLMENT (StdNo)
      ;
CREATE INDEX FK_COURSE
  on OFFERING (CourseNo)
      ;
CREATE INDEX FK_FACULTY
  on OFFERING (FacNo)
      ;
CREATE INDEX FK_FACSUPERVISOR
  on FACULTY (FacSupervisor)
      ;
/*--------------------------------------------------------------------------*/
/*    CREATE VIEWS                                                          */
/*--------------------------------------------------------------------------*/
CREATE VIEW STUDENT_VIEW as
     select StdFirstName, StdLastName, StdMajor
      from STUDENT;
       ;
```

```
create view two_table as
     select s.stdLastName, s.StdFirstName, S.Stdcity, s.StdGPA,
            e.enrgrade
     from student s, enrollment e
     where
     s.StdNo=e.StdNo
     ;
/*-------------------------------------------------------------------------------*/
/*    CHECK CONTENT OF CATALOG                                                   */
/*-------------------------------------------------------------------------------*/
SELECT SUBSTR(OBJECT_NAME, 1, 20), OBJECT_TYPE, STATUS
     from USER_OBJECTS
      ;
SPOOL OFF
```

Deliverable 3 - DDL

- ★ A DDL scripts with all the statements to create your databases and results of your query against the catalog.

Script Name: TeamName_ddl.sql

- ★ A log file displaying the output of the script.

Script name: TeamName_ddl.log

Task 4 - Load data

Now that 5 tables have been created, it's time to put some data into each table. Since you are using declarative referential integrity, the order of data input is important. Put data into the "one" sided tables first. The command used to put the data into the table is called INSERT. It is a standard command in industry.

Lab Assignment for Task 4

1. Set up a script that will load data into your five tables.
   Script name: TeamName_loadata.sql
2. Spool the output to a file.
   Script name: TeamName_loadata.log
3. Execute the script in SQL*Plus.
4. Debug the script.

Checklist

❖ *Set up your script to load the data.*

- Open up a text editor (notepad) and save with filename TeamName_loadata.sql.
- On the first line, type the spool command with the name of the file (TeamName_loadata.log).
- Write 5 SQL statements using the INSERT command to load 5 rows into each of your tables. So you will have a total of 25 insert statements to load data into your 5 tables.

★DBA Tip!

★ Pay attention to the data types you defined for the columns of your tables. If a column was defined as CHAR or VARCHAR2 when the table was created, add single quotation marks around the value of the data in the script. If the data was defined as NUMBER, then no quotation marks are required.

★ Attention also has to be given to referential integrity when data is inserted into tables. Therefore, data will be inserted into the "one" sided tables first. Then data could be inserted into the "many" sided tables. In all cases, the foreign key columns on the "many" sided tables need to have data that match on a "one" sided table.

Debug DML Script

Your DML script just like your DDL script may not run correctly the first time. The spooled file can be used to debug the script.

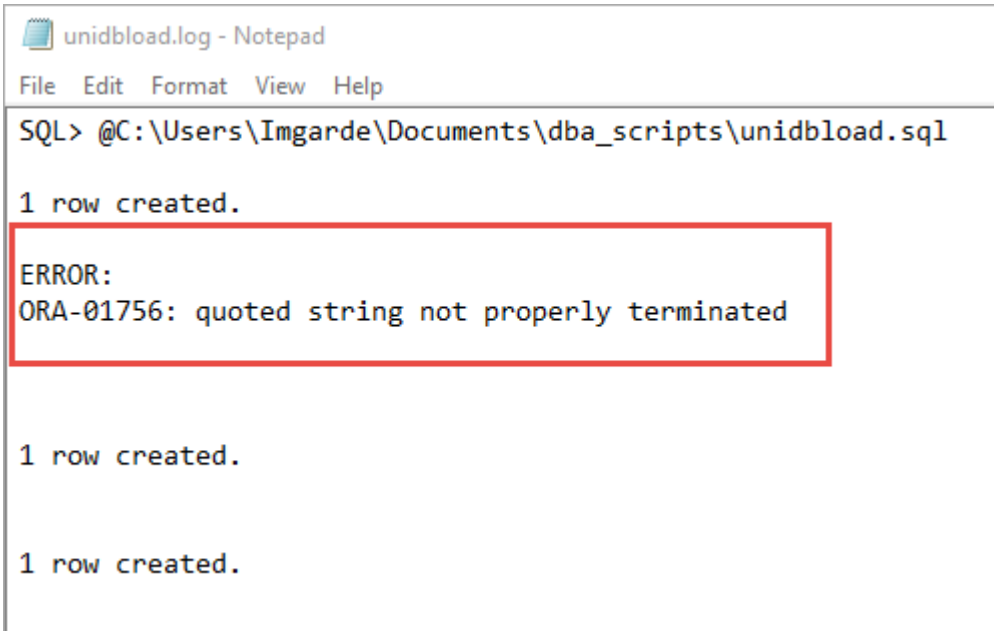
Guidelines for debugging scripts

Always start at the top of the spooled log file and look at the first error. Debug the script to load data you just created above.

1. **Formatting errors** - some of the errors in the logs from the INSERT statements are due to formatting errors.

   For example:

   ERROR:
   ORA-01756: quoted string not properly terminated



Notice on the row below that a double quote rather than a single quotation mark caused the error.

```
INSERT INTO student
        (stdNo, stdFirstName, stdLastName, stdCity,
         stdState, stdMajor, stdClass, stdGPA, stdZip)
        VALUES ('124-56-
7890','BOB','NORBERT','BOTHELL','WA','FIN','JR',2.70,'98011-2121'');
```

You will fix the insert statement by removing the double quotes and replacing with single quotes as highlighted below.

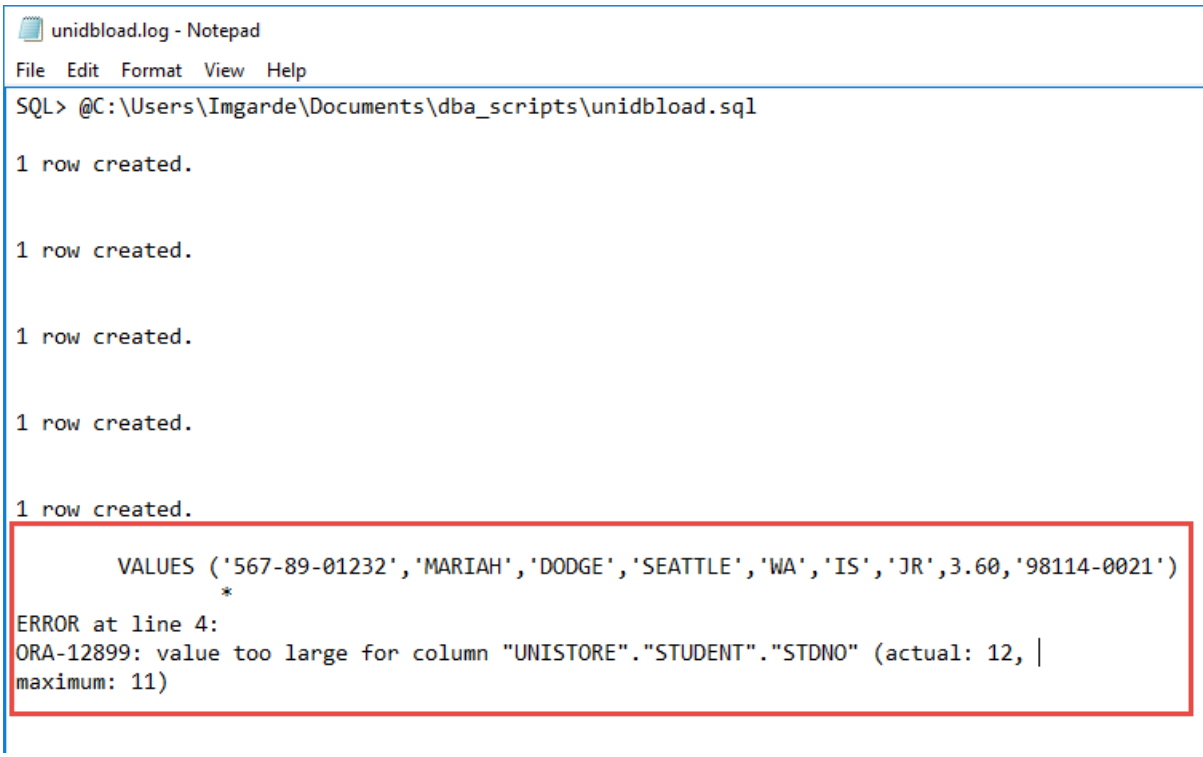
```
INSERT INTO student
        (stdNo, stdFirstName, stdLastName, stdCity,
         stdState, stdMajor, stdClass, stdGPA, stdZip)
        VALUES ('124-56-
7890','BOB','NORBERT','BOTHELL','WA','FIN','JR',2.70,'98011-2121');
```

2. Constraints on columns

The following error violates constraints on one of the rows in the StdNo column on the Student table. In the CREATE TABLE statement, the data type for the StdNo column is defined as CHAR with a fixed length of 11.

In the example below, the value for the StdNo entered is 12 characters long (567-89-01232) while the maximum acceptable value is 11.

```
INSERT INTO student
(stdNo, stdFirstName, stdLastName, stdCity,
 stdState, stdMajor, stdClass, stdGPA, stdZip)
VALUES ('567-89-
01232','MARIAH','DODGE','SEATTLE','WA','IS','JR',3.60,'98114-0021');
ERROR at line 4:
ORA-12899: value too large for column
"UNISTORE"."STUDENT"."STDNO" (actual: 12, maximum: 11)
```



To fix the problem, the value has to be made consistent with the data type specified in the CREATE TABLE statement.

```
INSERT INTO student
(stdNo, stdFirstName, stdLastName, stdCity,
stdState, stdMajor, stdClass, stdGPA, stdZip)
VALUES ('567-89-
0123','MARIAH','DODGE','SEATTLE','WA','IS','JR',3.60,'98114-0021');
```

3. Constraint errors

Some errors can be caused by violations of the primary/foreign key constraints. The error in the spool output file follows:

```
INSERT INTO enrollment
*
ERROR at line 1:
ORA-02291: integrity constraint (UNISTORE.STUDENTFK) violated
- parent key not
found
```

The full insert statement that caused the problem is:

```
INSERT INTO enrollment
(OfferNO, StdNo, EnrGrade)
VALUES(1234,'123-45-6789',3.30);
```

Notice that the value for the StdNo that is being inserted into the Enrolment table is **123-45-6789**. There is no StdNo on the Student table with the value **123-45-6789** that matches this value to be inserted into the enrollment table.

The value may not have been inserted due to formatting errors and the database is doing exactly what it should; it is protecting the database by preventing the entry of invalid or "bad" data.

To correct the data, check if the value for the StdNo entered exist or of it was incorrectly entered and the fix it. This error is not as easy to resolve as some errors, but it is crucial to have correct data in the database.

Sample Solution!

```
SPOOL TeamName_loadata.log

DELETE FROM ENROLLMENT;
DELETE FROM OFFERING;
DELETE FROM STUDENT;
DELETE FROM COURSE;
DELETE FROM FACULTY;
INSERT INTO student
      (stdNo, stdFirstName, stdLastName, stdCity, stdState, stdMajor,
       stdClass, stdGPA, stdZip)VALUES ('123-45-
6789','HOMER','WELLS','SEATTLE','WA','IS','FR',3.00,'98121-1111');
INSERT INTO student
      (stdNo, stdFirstName, stdLastName, stdCity, stdState, stdMajor,
       stdClass, stdGPA, stdZip)VALUES ('124-56-
7890','BOB','NORBERT','BOTHELL','WA','FIN','JR',2.70,'98011-2121');
INSERT INTO student
      (stdNo, stdFirstName, stdLastName, stdCity,stdState, stdMajor,
       stdClass, stdGPA, stdZip) VALUES ('234-56-
7890','CANDY','KENDALL','TACOMA','WA','ACCT','JR',3.50,'99042-3321');
INSERT INTO student
      (stdNo, stdFirstName, stdLastName, stdCity, stdState, stdMajor,
       stdClass, stdGPA, stdZip)VALUES ('345-67-
8901','WALLY','KENDALL','SEATTLE','WA','IS','SR',2.80,'98123-1141');
INSERT INTO student
      (stdNo, stdFirstName, stdLastName, stdCity, stdState, stdMajor,
       stdClass, stdGPA, stdZip) VALUES ('456-78-
9012','JOE','ESTRADA','SEATTLE','WA','FIN','SR',3.20,'98121-2333');
INSERT INTO course
      (CourseNo, crsDesc, CrsUnits)VALUES ( 'FIN300','FUNDAMENTALS OF
FINANCE',4);
INSERT INTO course
      (CourseNo, crsDesc, CrsUnits)VALUES ( 'FIN450','PRINCIPLES OF
INVESTMENTS',4);
INSERT INTO course (CourseNo, crsDesc, CrsUnits)VALUES (
'FIN480','CORPORATE FINANCE',4);
INSERT INTO course
        (CourseNo, crsDesc, CrsUnits) VALUES ('IS320','FUNDAMENTALS OF
BUSINESS PROGRAMMING',4 );
INSERT INTO course
      (CourseNo, crsDesc, CrsUnits)VALUES ( 'IS460','SYSTEMS ANALYSIS',4);
INSERT INTO faculty
      (FacNo, FacFirstName, FacLastName, FacCity, FacState, FacDept,
FacRank, FacSalary, FacSupervisor, FacHireDate, FacZipCode) VALUES ('543-
21-0987','VICTORIA','EMMANUEL','BOTHELL','WA','MS','PROF',
              120000.0,'','15-Apr-2001','98011-2242');
INSERT INTO faculty
      (FacNo, FacFirstName, FacLastName, FacCity, FacState, FacDept,
       FacRank, FacSalary, FacSupervisor, FacHireDate, FacZipCode)
       VALUES ('765-43-2109','NICKI','MACON','BELLEVUE','WA','FIN','PROF',
65000.00,'', '11-Apr-2002','98015-9945');

INSERT INTO faculty
      (FacNo, FacFirstName, FacLastName, FacCity, FacState, FacDept,
```

```
       FacRank, FacSalary, FacSupervisor, FacHireDate, FacZipCode)
       VALUES ('654-32-1098','LEONARD','FIBON','SEATTLE','WA','MS','ASSC',
              70000.00,'543-21-0987','01-May-1999','98121-0094');
INSERT INTO faculty
      (FacNo, FacFirstName, FacLastName, FacCity, FacState, FacDept,
       FacRank, FacSalary, FacSupervisor, FacHireDate, FacZipCode)
       VALUES ('098-76-
5432','LEONARD','VINCE','SEATTLE','WA','MS','ASST',35000.00,
            '654-32-1098','10-Apr-2000','98111-9921');
INSERT INTO faculty
      (FacNo, FacFirstName, FacLastName, FacCity, FacState, FacDept,
       FacRank, FacSalary, FacSupervisor, FacHireDate, FacZipCode)
       VALUES ('876-54-
3210','CRISTOPHER','COLAN','SEATTLE','WA','MS','ASST',
              40000.00,'654-32-1098','01-Mar-2004','98114-1332');
INSERT INTO offering
      (OfferNo, CourseNo, OffTerm, OffYear, OffLocation, OffTime, FacNo,
       OffDays)
       VALUES (1111,'IS320','SUMMER',2013,'BLM302','10:30:00',NULL,'MW');
INSERT INTO offering
      (OfferNo, CourseNo, OffTerm, OffYear, OffLocation, OffTime, FacNo,
       OffDays)
       VALUES (1234,'IS320','FALL',2012,'BLM302','10:30:00','098-76-
5432','MW');
INSERT INTO offering
      (OfferNo, CourseNo, OffTerm, OffYear, OffLocation, OffTime, FacNo,
       OffDays)
       VALUES (2222,'IS460','SUMMER',2012,'BLM412','13:30:00',NULL,'TTH');
INSERT INTO offering
      (OfferNo, CourseNo, OffTerm, OffYear, OffLocation, OffTime, FacNo,
       OffDays)
       VALUES (3333,'IS320','SPRING',2013,'BLM214','08:30:00','098-76-
5432','MW');
INSERT INTO offering
      (OfferNo, CourseNo, OffTerm, OffYear, OffLocation, OffTime, FacNo,
       OffDays)VALUES (4321,'IS320','FALL',2012,'BLM214','15:30:00','098-
76-5432','TTH');
INSERT INTO enrollment
      (OfferNO, StdNo, EnrGrade)VALUES(1234,'123-45-6789',3.30);
INSERT INTO enrollment
      (OfferNO, StdNo, EnrGrade) VALUES(1234,'234-56-7890',3.50);
INSERT INTO enrollment
      (OfferNO, StdNo, EnrGrade) VALUES(1234,'345-67-8901',3.20);
INSERT INTO enrollment
      (OfferNO, StdNo, EnrGrade)VALUES(1234,'456-78-9012',3.10);
INSERT INTO enrollment
      (OfferNO, StdNo, EnrGrade)VALUES(1234,'124-56-7890',3.80);
commit;

SPOOL OFF
```

Deliverable 4 – DML

- ★ SQL script to load your data.

  Script Name: TeamName_loadata.sql
- ★ Spooled log file.

  Script Name: TeamName_loadata.log

Task 5 - Retrieve the Data

You loaded data in each of your five tables in Task 4. This data is now available for your users to retrieve for their business purposes. Users can query the database using SQL statements to retrieve information from specific columns in individual tables, all columns in individual tables or from multiple tables.

You will use SQL statements to display data from your database tables.

Lab Assignment for Task 5

1. Set up a script that will retrieve data from the tables.
   Script Name: TeamName_retreievedata.sql
2. The data retrieved should meet the following constraints:
   a. Using a single query select all the rows of one table.
   b. Using a single query select some rows of one table.
   c. Using a single query select data from a table that meets specific conditions.
   d. Use the IN statement to filter rows.
   e. Using an equijoin view perform a two table join.
   f. Using an equijoin view perform a three table join.
   g. Create a table like one of your tables. Alter it and add a column.
3. Use the spool tool to capture the output to a spool file.
   Name of spooled file: TeamName_retrievedata.log
4. Execute the script in SQL*Plus

Checklist

- Decide the information you want to retrieve and then determine which tables contain this information. Write it down.
- Create a query using the SQL SELECT statement including the columns you want to display followed by the FROM keyword and the name of the table(s) the information resides on.
- A simple query to return all the columns and rows from the STUDENT table will look like this:

```
SELECT * FROM STUDENT;
```

- To retrieve information from specific columns in a table, you will modify the query to look like the following:

```
SELECT StdNO, StdFirstName, StdLastName FROM STUDENT;
```

This query will retrieve the student number, first name and last name of all the students enrolled in the university.

- To retrieve data to meet a certain criteria, use the WHERE clause of the SQL statement to limit the rows returned to meet your criteria. For example

```
SELECT * FROM STUDENT
WHERE STDLASTNAME = 'KENDALL';
```

- To retrieve data from multiple tables, you will use a JOIN. Usually tables are "joined" on the primary and foreign keys. That's why the relationships that were specified in the ERD are so important. When a "join" is needed, the WHERE keyword is again used and an equal condition is specified. That's why the join is called an "equijoin".

  For example:

```
SELECT STDLAST_NAME, STDCITY, STDSTATE, ENRGRADE, OFFERNO
FROM STUDENT, ENROLLMENT
WHERE STUDENT.STDNO = ENROLLMENT.STDNO;
```

This query will list the last names, city and state of all the students in the Student table and their offerNo and Enrollment grade.

Notice that even though the tables are joined on the StdNo column, the StdNo will not be displayed.

The result is just the 5 students that were inserted earlier in the Student table. The OfferNo and Engrade values that were inserted into the Enrollment table and that had a matching foreign key on the Student table will be displayed with the corresponding student name.

The example below provides details of offerings and assigned faculty for fall 2012 IS courses taught by assistant professors

```
SELECT OfferNo, CourseNo, FacFirstName, FacLastName
FROM Offering, Faculty
WHERE OffTerm = 'FALL' AND OffYear = 2012
AND FacRank = 'ASST' AND CourseNo LIKE 'IS%'
AND Faculty.FacNo = Offering.FacNo;
```

- Sometimes an attribute with the same name exists on several tables. Or sometimes a table name is long and the programmer doesn't want to have to type the long name in each time. In these cases, a surrogate can be used to represent the names of the tables. The surrogate is usually the first initial of the table name. In the query that follows, the letter F denotes the FACULTY table and the letter O denotes the OFFERING table.

```
SELECT * FROM FACULTY C, OFFERING O
WHERE F.FacNo=O.FacNo
AND F.FacLastName='EMMANUEL';
```

This query also has another condition that must be satisfied; the last name must be Emmanuel. So, two conditions must be met; the faculty numbers must be equal and the last name must be Emmanuel.

Sample Solution!

```
SPOOL TeamName_retrievedata.log

SELECT * FROM STUDENT;

SELECT StFirstName,StdLastName, StdState, StdMajor from STUDENT;

SELECT * FROM STUDENT
WHERE StdlastName = 'KENDALL';

SELECT STDLASTNAME, STDCITY, STDSTATE, ENRGRADE, OFFERNO
FROM STUDENT, ENROLLMENT
WHERE STUDENT.STDNO = ENROLLMENT.STDNO;

SELECT * FROM FACULTY F, OFFERING O
WHERE F.FacNo=O.FacNo
AND F.FacLastName='EMMANUEL';

SELECT OfferNo, CourseNo, FacFirstName, FacLastName
FROM Offering, Faculty
WHERE OffTerm = 'FALL' AND OffYear = 2012
AND FacRank = 'ASST' AND CourseNo LIKE 'IS%'
AND Faculty.FacNo = Offering.FacNo;


SPOOL OFF
```

Final Deliverable

★ Data retrieval script and log file.log.

Script Name: TeamName_retrievedata.sql

File name: TeamName_retrievedata.log

★ A word document including the following SQL scripts.

- Statement of work
- ERD
- DDL script including the log files
- Load script including the log files
- Data retrieval script including the log files

★ **All the log files should be added as an appendix at the end of your final word document and should be properly labeled**

Deliverables Due Dates

Deliverable	Expected Due Date
SOW	03/09/17
ERD	03/9/17
DDL	03/16/17
DML	03/16/17
FINAL	03/23/17

Note: Following directions are CRITICAL for this exercise and points will be deducted if all the files are not submitted with proper naming conventions. You will earn a total of 100 points for this assignment.

.