Instructor Teaching Tracker: Security Project

Brandon R. Russell

University of Maryland University College

Abstract

This paper provides an

Keywords: Database, Instructor, Teaching, Tracker, Security

Contents

[Abstract 2](#_Toc526670327)

[Instructor Teaching Tracker: Security Project 6](#_Toc526670328)

[Statement of Work 6](#_Toc526670329)

[Database Name 6](#_Toc526670330)

[Database Objective 6](#_Toc526670331)

[The Problem 7](#_Toc526670332)

[Security Plan 7](#_Toc526670333)

[Technical Details 8](#_Toc526670334)

[Timeline 8](#_Toc526670335)

[Data Model 9](#_Toc526670336)

[Conceptual 9](#_Toc526670337)

[User Roles. 9](#_Toc526670338)

[Assumptions. 9](#_Toc526670339)

[Business Rules. 9](#_Toc526670340)

[Logical 10](#_Toc526670341)

[Implementation 11](#_Toc526670342)

[DDL 11](#_Toc526670343)

[DML 18](#_Toc526670344)

[Security 34](#_Toc526670345)

[Policies 34](#_Toc526670346)

[Password Policy. 35](#_Toc526670347)

[Connection Policy. 35](#_Toc526670348)

[Role Assignment Policy. 35](#_Toc526670349)

[Account Modify Policy. 35](#_Toc526670350)

[Course Management Policy. 35](#_Toc526670351)

[Section Management Policy. 35](#_Toc526670352)

[Class Schedule Modify Policy. 35](#_Toc526670353)

[Admin Support Policy. 35](#_Toc526670354)

[Instructor View Policy. 35](#_Toc526670355)

[Instructor Modify Policy. 36](#_Toc526670356)

[Student View Policy. 36](#_Toc526670357)

[Student Modify Policy. 36](#_Toc526670358)

[Procedures 36](#_Toc526670359)

[Password Procedure. 36](#_Toc526670360)

[Connection Procedure. 36](#_Toc526670361)

[Role Assignment Procedure. 36](#_Toc526670362)

[Account Modify Procedure. 36](#_Toc526670363)

[Course Management Procedure. 36](#_Toc526670364)

[Section Management Procedure. 36](#_Toc526670365)

[Class Schedule Modify Procedure. 37](#_Toc526670366)

[Admin Support Procedure. 37](#_Toc526670367)

[Instructor View Procedure. 37](#_Toc526670368)

[Instructor Modify Procedure. 37](#_Toc526670369)

[Student View Procedure. 37](#_Toc526670370)

[Student Modify Procedure. 37](#_Toc526670371)

[Conclusion 37](#_Toc526670372)

[Lesson Learned 37](#_Toc526670373)

[Final Thoughts 37](#_Toc526670374)

[References 38](#_Toc526670375)

Instructor Teaching Tracker: Security Project

This

# Statement of Work

## **Database Name**

Instructor Teaching Tracker

Database Objective

The purpose of this project is to implement security measures into the existing instructor teaching tracker database. This database supports the Instructor Teaching Tracker application for Training Detachment 6. It keeps track of instructor, course, admin, student, and section data; additionally, it tracks classes taught, the schedule, and students signed up for classes. The overall purpose is assisting instructors, administrators, and students with creating, scheduling, maintaining, and monitoring personal, course, and class data in an efficient manner. The instructor, administrator, and student tables hold all pertinent personal information, such as: name, address, and phone number. The course list table is a comprehensive listing of all courses taught across the different sections of the school, and it contains the name, author, description, hours, and section number for each course. The section table provides basic information on the sections which instructors and courses belong to, and it holds the following information: section number, name, address, and field of study. The class schedule table holds the different schedules classes can follow and gives the day and time a class is taught, a down day schedule, and notes. The Instructor Classes table is at the core of the database, providing a complete schedule of all courses taught by all instructors, with schedules, and also the room number of the class and any instructor notes. The Student Class Signup table provides students the ability to sign up for available classes in the Instructor Classes table.

## The Problem

Training Detachment 6 is a military field training facility, employing eleven instructors and eleven support admin staff at any given time, and host a variable number of students annually. The teaching tracker application and associated database became a necessity due to the complexities of tracking a large number of instructors, courses, and students. The database was created in house, and as a result no security was implemented within the initial design. In an effort to maintain the confidentiality of certain data, such as instructor’s phone numbers and addresses, and to ensure the integrity of the data, i.e. prevent students from changing course data, we need to hire a security consultant to redesign the database with security permissions added.

# Security Plan

One of the most important steps in this project is to create a sound security plan. This document will be an overarching guide on set requirements to be enforced through security policies to establish sound security within the database (Theriault & Heney, 1998). For our database, the Instructor Teaching Tracker, we will achieve the necessary security by separating user accounts into specific groups, i.e. students, instructors, and support administrators. We will then enact policies to assign roles to these groups with specific actions they are allowed to use, and this will be augmented with Views and Oracle Label Security. There will also be established policies on standard password requirements, and the enforcement of autogenerated identification numbers to reduce the storing of sensitive data. All security policies will have an accompanying procedure for implementation, and this will be thoroughly tested and documented before operationalized. Any user in violation of this plan or subsequent policies are subject to account disablement.

# Technical Details

The database was built with Oracle 11g on a Windows platform. We used Oracle SQL Developer Data Modeler to design the Entity Relationship Diagram (ERD). All Data Definition Language (DDL), Data Manipulation Language (DML), and Data Control Language (DCL) implementation scripts were rendered in Oracle SQL Developer or SQL Plus.

# Timeline

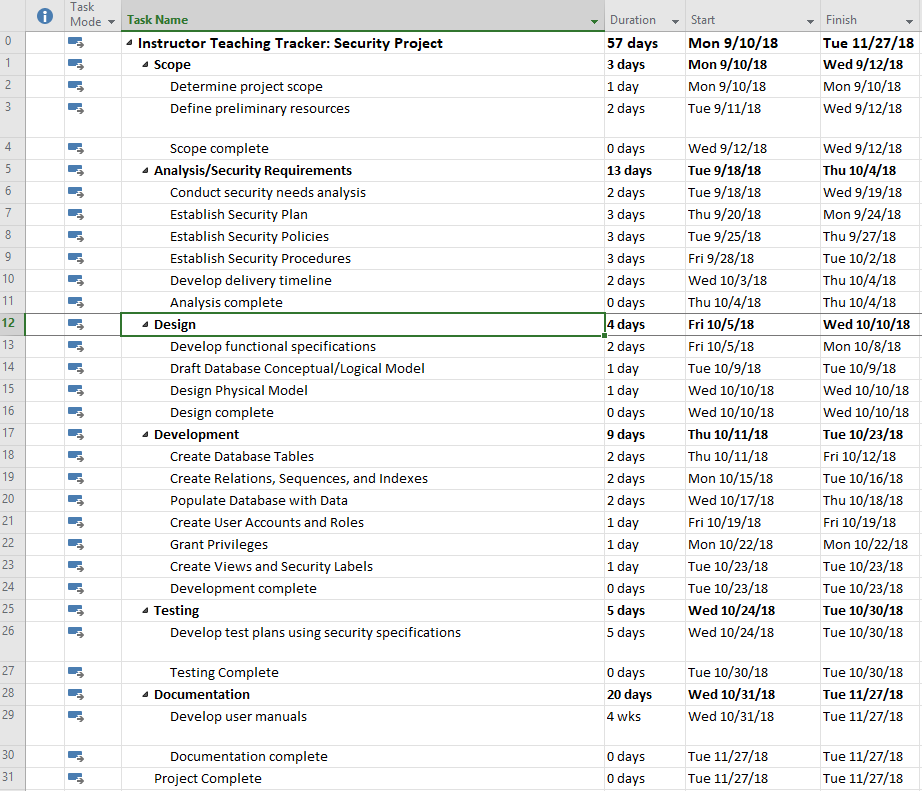


Figure 1. Project timeline created with Microsoft Project.

# Data Model

## Conceptual

User Roles. There will be three different primary user roles: student, instructor, and support admin. Specific policies will be defined in the security policy section for how each user role can and cannot interact with the database.

Assumptions. This application tracks classes taught by instructors and includes a class schedule. It is assumed classes will always meet a specific schedule criteria, i.e. Monday-Wednesday-Friday, Tuesday-Thursday, or Saturday. Down days are tracked for individual classes since this will vary based upon the specific class. For the usernames, we are using the user’s first character of their first name concatenated with the full last name with their sequence id added to the end. There is still a chance of duplicate user names since instructor, student, and admin information is stored in different tables, but for the purpose of this project we are assuming this will not occur due to the low number of users.

Business Rules. The business rules to follow for this database are as follows:

1. A section may hire zero or more instructors, but an instructor must belong to only one section.
2. A section may own zero or more courses, but a course must be owned by only one section.
3. A section may hire zero or more administrators, but an administrator must belong to only one section.
4. An administrator can own multiple courses, a course can only be owned by one administrator.
5. A course may generate zero or more classes, but a class can only be generated from one course.
6. An instructor may teach zero or more classes, but a class can only be taught by one instructor.
7. A schedule may be generated for zero or more classes, but a class must only have one schedule.
8. A class may have zero or more students.
9. A student can sign up for zero or more classes, so long as there is no schedule conflict.

## Logical

Figure 2 provides a logical rendering of the database design, while figure 3 provides a more detailed ERD of the database design.

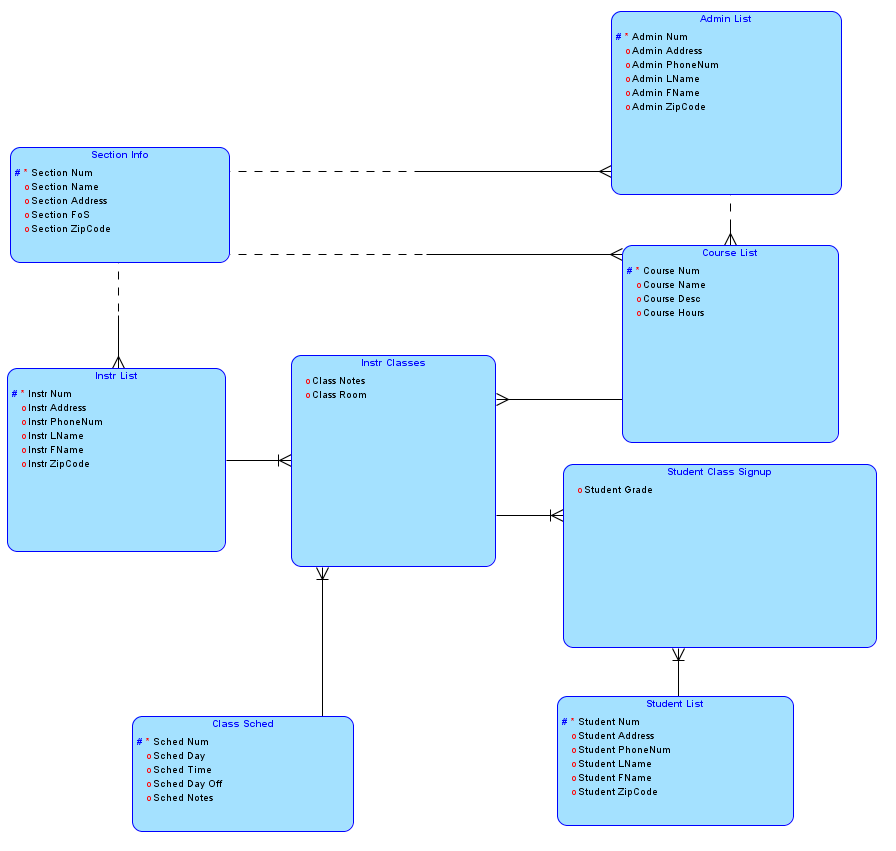


Figure 2. Logical model created with Oracle SQL Developer Data Modeler.

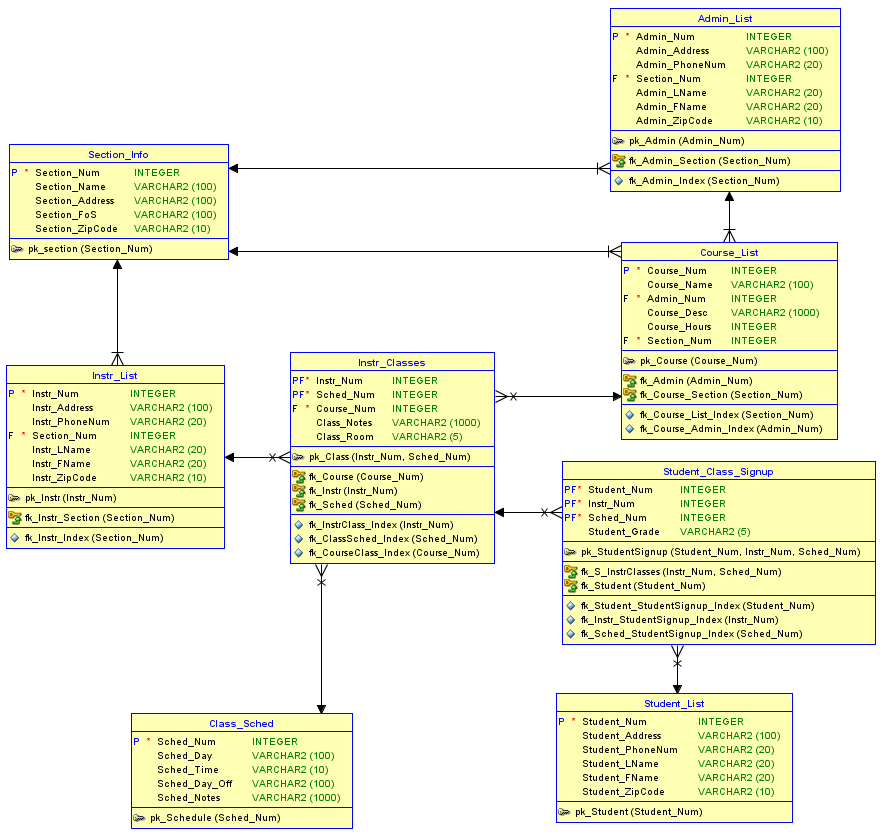


Figure 3. ERD created with Oracle SQL Developer Data Modeler.

# Implementation

This section consists of the physical implementation of the database, i.e. DDLs, used to create the database and the DMLs used to insert data into it.

## DDL

SQL> SET echo on;

SQL> SET serveroutput on;

SQL> /\* Drop tables, sequence, and other objects you create\*/

SQL>

SQL> DROP TABLE Student\_Class\_Signup;

Table STUDENT\_CLASS\_SIGNUP dropped.

SQL> DROP TABLE Instr\_Classes;

Table INSTR\_CLASSES dropped.

SQL> DROP TABLE Student\_List;

Table STUDENT\_LIST dropped.

SQL> DROP TABLE Instr\_List;

Table INSTR\_LIST dropped.

SQL> DROP TABLE Course\_List;

Table COURSE\_LIST dropped.

SQL> DROP TABLE Admin\_List;

Table ADMIN\_LIST dropped.

SQL> DROP TABLE Section\_Info;

Table SECTION\_INFO dropped.

SQL> DROP TABLE Class\_Sched;

Table CLASS\_SCHED dropped.

SQL> DROP SEQUENCE SectionNum\_Seq;

Sequence SECTIONNUM\_SEQ dropped.

SQL> DROP SEQUENCE InstrNum\_Seq;

Sequence INSTRNUM\_SEQ dropped.

SQL> DROP SEQUENCE CourseNum\_Seq;

Sequence COURSENUM\_SEQ dropped.

SQL> DROP SEQUENCE AdminNum\_Seq;

Sequence ADMINNUM\_SEQ dropped.

SQL> DROP SEQUENCE SchedNum\_Seq;

Sequence SCHEDNUM\_SEQ dropped.

SQL> DROP SEQUENCE StudentNum\_Seq;

Sequence STUDENTNUM\_SEQ dropped.

SQL>

SQL> /\* Create tables \*/

SQL>

SQL> CREATE TABLE Section\_Info

2 (

3 Section\_Num INTEGER NOT NULL,

4 Section\_Name VARCHAR (100),

5 Section\_Address VARCHAR (100),

6 Section\_FoS VARCHAR (100),

7 Section\_ZipCode VARCHAR(10),

8 CONSTRAINT pk\_section PRIMARY KEY (Section\_Num)

9 );

Table SECTION\_INFO created.

SQL>

SQL> DESCRIBE Section\_Info;

Name Null? Type

--------------- -------- -------------

SECTION\_NUM NOT NULL NUMBER(38)

SECTION\_NAME VARCHAR2(100)

SECTION\_ADDRESS VARCHAR2(100)

SECTION\_FOS VARCHAR2(100)

SECTION\_ZIPCODE VARCHAR2(10)

SQL>

SQL> CREATE TABLE Admin\_List

2 (

3 Admin\_Num INTEGER NOT NULL,

4 Admin\_Address VARCHAR (100),

5 Admin\_PhoneNum VARCHAR (20),

6 Section\_Num INTEGER NOT NULL,

7 Admin\_LName VARCHAR (20),

8 Admin\_FName VARCHAR (20),

9 Admin\_ZipCode VARCHAR (10),

10 CONSTRAINT pk\_Admin PRIMARY KEY (Admin\_Num),

11 CONSTRAINT fk\_Admin\_Section FOREIGN KEY (Section\_Num)

12 REFERENCES Section\_Info

13 );

Table ADMIN\_LIST created.

SQL>

SQL> DESCRIBE Admin\_List;

Name Null? Type

-------------- -------- -------------

ADMIN\_NUM NOT NULL NUMBER(38)

ADMIN\_ADDRESS VARCHAR2(100)

ADMIN\_PHONENUM VARCHAR2(20)

SECTION\_NUM NOT NULL NUMBER(38)

ADMIN\_LNAME VARCHAR2(20)

ADMIN\_FNAME VARCHAR2(20)

ADMIN\_ZIPCODE VARCHAR2(10)

SQL>

SQL>

SQL> CREATE TABLE Student\_List

2 (

3 Student\_Num INTEGER NOT NULL,

4 Student\_Address VARCHAR (100),

5 Student\_PhoneNum VARCHAR (20),

6 Student\_LName VARCHAR (20),

7 Student\_FName VARCHAR (20),

8 Student\_ZipCode VARCHAR (10),

9 CONSTRAINT pk\_Student PRIMARY KEY (Student\_Num)

10 );

Table STUDENT\_LIST created.

SQL>

SQL> DESCRIBE Student\_List;

Name Null? Type

---------------- -------- -------------

STUDENT\_NUM NOT NULL NUMBER(38)

STUDENT\_ADDRESS VARCHAR2(100)

STUDENT\_PHONENUM VARCHAR2(20)

STUDENT\_LNAME VARCHAR2(20)

STUDENT\_FNAME VARCHAR2(20)

STUDENT\_ZIPCODE VARCHAR2(10)

SQL>

SQL>

SQL> CREATE TABLE Instr\_List

2 (

3 Instr\_Num INTEGER NOT NULL,

4 Instr\_Address VARCHAR (100),

5 Instr\_PhoneNum VARCHAR (20),

6 Section\_Num INTEGER NOT NULL,

7 Instr\_LName VARCHAR (20),

8 Instr\_FName VARCHAR (20),

9 Instr\_ZipCode VARCHAR (10),

10 CONSTRAINT pk\_Instr PRIMARY KEY (Instr\_Num),

11 CONSTRAINT fk\_Instr\_Section FOREIGN KEY (Section\_Num)

12 REFERENCES Section\_Info

13 );

Table INSTR\_LIST created.

SQL>

SQL> DESCRIBE Instr\_List;

Name Null? Type

-------------- -------- -------------

INSTR\_NUM NOT NULL NUMBER(38)

INSTR\_ADDRESS VARCHAR2(100)

INSTR\_PHONENUM VARCHAR2(20)

SECTION\_NUM NOT NULL NUMBER(38)

INSTR\_LNAME VARCHAR2(20)

INSTR\_FNAME VARCHAR2(20)

INSTR\_ZIPCODE VARCHAR2(10)

SQL>

SQL>

SQL> CREATE TABLE Course\_List

2 (

3 Course\_Num INTEGER NOT NULL,

4 Course\_Name VARCHAR (100),

5 Admin\_Num INTEGER NOT NULL,

6 Course\_Desc VARCHAR (1000),

7 Course\_Hours INTEGER,

8 Section\_Num INTEGER NOT NULL,

9 CONSTRAINT pk\_Course PRIMARY KEY (Course\_Num),

10 CONSTRAINT fk\_Course\_Section FOREIGN KEY (Section\_Num)

11 REFERENCES Section\_Info,

12 CONSTRAINT fk\_Admin FOREIGN KEY (Admin\_Num)

13 REFERENCES Admin\_List

14

15 );

Table COURSE\_LIST created.

SQL>

SQL> DESCRIBE Course\_List;

Name Null? Type

------------ -------- --------------

COURSE\_NUM NOT NULL NUMBER(38)

COURSE\_NAME VARCHAR2(100)

ADMIN\_NUM NOT NULL NUMBER(38)

COURSE\_DESC VARCHAR2(1000)

COURSE\_HOURS NUMBER(38)

SECTION\_NUM NOT NULL NUMBER(38)

SQL>

SQL>

SQL> CREATE TABLE Class\_Sched

2 (

3 Sched\_Num INTEGER NOT NULL,

4 Sched\_Day VARCHAR (100),

5 Sched\_Time VARCHAR (10),

6 Sched\_Day\_Off VARCHAR (100),

7 Sched\_Notes VARCHAR (1000),

8 CONSTRAINT pk\_Schedule PRIMARY KEY (Sched\_Num)

9 );

Table CLASS\_SCHED created.

SQL>

SQL> DESCRIBE Class\_Sched;

Name Null? Type

------------- -------- --------------

SCHED\_NUM NOT NULL NUMBER(38)

SCHED\_DAY VARCHAR2(100)

SCHED\_TIME VARCHAR2(10)

SCHED\_DAY\_OFF VARCHAR2(100)

SCHED\_NOTES VARCHAR2(1000)

SQL>

SQL> CREATE TABLE Instr\_Classes

2 (

3

4 Instr\_Num INTEGER NOT NULL,

5 Sched\_Num INTEGER NOT NULL,

6 Course\_Num INTEGER NOT NULL,

7 Class\_Notes VARCHAR (1000),

8 Class\_Room VARCHAR (5),

9 CONSTRAINT pk\_Class PRIMARY KEY (Instr\_Num, Sched\_Num),

10 CONSTRAINT fk\_Instr FOREIGN KEY (Instr\_Num)

11 REFERENCES Instr\_List

12 ON DELETE CASCADE,

13 CONSTRAINT fk\_Sched FOREIGN KEY (Sched\_Num)

14 REFERENCES Class\_Sched

15 ON DELETE CASCADE,

16 CONSTRAINT fk\_Course FOREIGN KEY (Course\_Num)

17 REFERENCES Course\_List

18 ON DELETE CASCADE

19 );

Table INSTR\_CLASSES created.

SQL>

SQL> DESCRIBE Instr\_Classes;

Name Null? Type

----------- -------- --------------

INSTR\_NUM NOT NULL NUMBER(38)

SCHED\_NUM NOT NULL NUMBER(38)

COURSE\_NUM NOT NULL NUMBER(38)

CLASS\_NOTES VARCHAR2(1000)

CLASS\_ROOM VARCHAR2(5)

SQL>

SQL>

SQL> CREATE TABLE Student\_Class\_Signup

2 (

3 Student\_Num INTEGER NOT NULL,

4 Instr\_Num INTEGER NOT NULL,

5 Sched\_Num INTEGER NOT NULL,

6 Student\_Grade VARCHAR (5),

7 CONSTRAINT pk\_StudentSignup PRIMARY KEY (Student\_Num, Instr\_Num, Sched\_Num),

8 CONSTRAINT fk\_Student FOREIGN KEY (Student\_Num)

9 REFERENCES Student\_List

10 ON DELETE CASCADE,

11 CONSTRAINT fk\_S\_InstrClasses FOREIGN KEY (Instr\_Num, Sched\_Num)

12 REFERENCES Instr\_Classes(Instr\_Num, Sched\_Num)

13 ON DELETE CASCADE

14 );

Table STUDENT\_CLASS\_SIGNUP created.

SQL> /\* Create indexes on foreign keys\*/

SQL>

SQL> CREATE INDEX fk\_Course\_List\_Index on Course\_List(Section\_Num);

Index FK\_COURSE\_LIST\_INDEX created.

SQL> CREATE INDEX fk\_Course\_Admin\_Index on Course\_List(Admin\_Num);

Index FK\_COURSE\_ADMIN\_INDEX created.

SQL> CREATE INDEX fk\_Instr\_Index on Instr\_List(Section\_Num);

Index FK\_INSTR\_INDEX created.

SQL> CREATE INDEX fk\_Admin\_Index on Admin\_List(Section\_Num);

Index FK\_ADMIN\_INDEX created.

SQL> CREATE INDEX fk\_InstrClass\_Index on Instr\_Classes(Instr\_Num);

Index FK\_INSTRCLASS\_INDEX created.

SQL> CREATE INDEX fk\_ClassSched\_Index on Instr\_Classes(Sched\_Num);

Index FK\_CLASSSCHED\_INDEX created.

SQL> CREATE INDEX fk\_CourseClass\_Index on Instr\_Classes(Course\_Num);

Index FK\_COURSECLASS\_INDEX created.

SQL> CREATE INDEX fk\_Student\_StudentSignup\_Index on Student\_Class\_Signup(Student\_Num);

Index FK\_STUDENT\_STUDENTSIGNUP\_INDEX created.

SQL> CREATE INDEX fk\_Instr\_StudentSignup\_Index on Student\_Class\_Signup(Instr\_Num);

Index FK\_INSTR\_STUDENTSIGNUP\_INDEX created.

SQL> CREATE INDEX fk\_Sched\_StudentSignup\_Index on Student\_Class\_Signup(Sched\_Num);

Index FK\_SCHED\_STUDENTSIGNUP\_INDEX created.

SQL>

SQL> /\* Create trigger \*/

SQL> /\*This trigger will display a message when a row is added to Instr\_Classes\*/

SQL> CREATE OR REPLACE TRIGGER SchedClass\_Trigger AFTER INSERT ON Instr\_Classes

2 FOR EACH ROW

3 BEGIN

4 dbms\_output.put\_line ('A class has been added!!');

5 END;

6 /

Trigger SCHEDCLASS\_TRIGGER compiled

SQL> /\* Create sequence\*/

SQL>

SQL> CREATE SEQUENCE SectionNum\_Seq

2 START WITH 1

3 INCREMENT BY 1;

Sequence SECTIONNUM\_SEQ created.

SQL>

SQL> CREATE SEQUENCE AdminNum\_Seq

2 START WITH 1

3 INCREMENT BY 1;

Sequence ADMINNUM\_SEQ created.

SQL>

SQL> CREATE SEQUENCE InstrNum\_Seq

2 START WITH 1

3 INCREMENT BY 1;

Sequence INSTRNUM\_SEQ created.

SQL>

SQL> CREATE SEQUENCE CourseNum\_Seq

2 START WITH 1

3 INCREMENT BY 1;

Sequence COURSENUM\_SEQ created.

SQL>

SQL> CREATE SEQUENCE SchedNum\_Seq

2 START WITH 1

3 INCREMENT BY 1;

Sequence SCHEDNUM\_SEQ created.

SQL>

SQL> CREATE SEQUENCE StudentNum\_Seq

2 START WITH 1

3 INCREMENT BY 1;

Sequence STUDENTNUM\_SEQ created.

## DML

SQL>

SQL> /\* Insert 10 or more rows into each table \*/

SQL>

SQL>

SQL> INSERT INTO Section\_Info (Section\_Num, Section\_Name, Section\_Address, Section\_FoS, Section\_ZipCode)

2 VALUES (SectionNum\_Seq.NEXTVAL, 'Computer Science Section', '110 Finegand Place','Computer Science','31088');

1 row inserted.

SQL> INSERT INTO Section\_Info (Section\_Num, Section\_Name, Section\_Address, Section\_FoS,Section\_ZipCode)

2 VALUES (SectionNum\_Seq.NEXTVAL, 'Art Section', '111 Finegand Place','Art','31088');

1 row inserted.

SQL> INSERT INTO Section\_Info (Section\_Num, Section\_Name, Section\_Address, Section\_FoS,Section\_ZipCode)

2 VALUES (SectionNum\_Seq.NEXTVAL, 'Aircraft Section', '112 Finegand Place','Aircraft','31088');

1 row inserted.

SQL> INSERT INTO Section\_Info (Section\_Num, Section\_Name, Section\_Address, Section\_FoS,Section\_ZipCode)

2 VALUES (SectionNum\_Seq.NEXTVAL, 'Robotic Section', '113 Finegand Place','Robotics','31088');

1 row inserted.

SQL> INSERT INTO Section\_Info (Section\_Num, Section\_Name, Section\_Address, Section\_FoS, Section\_ZipCode)

2 VALUES (SectionNum\_Seq.NEXTVAL, 'Database Section', '114 Finegand Place','Database Technology','31088');

1 row inserted.

SQL> INSERT INTO Section\_Info (Section\_Num, Section\_Name, Section\_Address, Section\_FoS, Section\_ZipCode)

2 VALUES (SectionNum\_Seq.NEXTVAL, 'English Section', '115 Finegand Place','English language','31088');

1 row inserted.

SQL> INSERT INTO Section\_Info (Section\_Num, Section\_Name, Section\_Address, Section\_FoS, Section\_ZipCode)

2 VALUES (SectionNum\_Seq.NEXTVAL, 'Physics Section', '116 Finegand Place','Physics','31088');

1 row inserted.

SQL> INSERT INTO Section\_Info (Section\_Num, Section\_Name, Section\_Address, Section\_FoS,Section\_ZipCode)

2 VALUES (SectionNum\_Seq.NEXTVAL, 'Chemistry Section', '117 Finegand Place','Chemistry','31088');

1 row inserted.

SQL> INSERT INTO Section\_Info (Section\_Num, Section\_Name, Section\_Address, Section\_FoS, Section\_ZipCode)

2 VALUES (SectionNum\_Seq.NEXTVAL, 'Spanish Section', '118 Finegand Place','Spanish language','31088');

1 row inserted.

SQL> INSERT INTO Section\_Info (Section\_Num, Section\_Name, Section\_Address, Section\_FoS, Section\_ZipCode)

2 VALUES (SectionNum\_Seq.NEXTVAL, 'Automotive Section', '119 Finegand Place','Automotive Mechanics','31088');

1 row inserted.

SQL> INSERT INTO Section\_Info (Section\_Num, Section\_Name, Section\_Address, Section\_FoS, Section\_ZipCode)

2 VALUES (SectionNum\_Seq.NEXTVAL, 'Space Travel Section', '120 Finegand Place','Space Travel','31088');

1 row inserted.

SQL>

SQL> INSERT INTO Admin\_List (Admin\_Num, Admin\_Address, Admin\_PhoneNum, Section\_Num, Admin\_LName, Admin\_FName, Admin\_ZipCode)

2 VALUES (AdminNum\_Seq.NEXTVAL,'501 Orange Park','111-333-3434', '1', 'Evans', 'Bob', '31004');

1 row inserted.

SQL> INSERT INTO Admin\_List (Admin\_Num, Admin\_Address, Admin\_PhoneNum, Section\_Num, Admin\_LName, Admin\_FName, Admin\_ZipCode)

2 VALUES (AdminNum\_Seq.NEXTVAL,'502 Orange Park','111-333-3435', '2', 'Johnson', 'Mike', '31004');

1 row inserted.

SQL> INSERT INTO Admin\_List (Admin\_Num, Admin\_Address, Admin\_PhoneNum, Section\_Num, Admin\_LName, Admin\_FName, Admin\_ZipCode)

2 VALUES (AdminNum\_Seq.NEXTVAL,'503 Orange Park','111-333-3436', '3', 'Jenkins', 'Mary', '31004');

1 row inserted.

SQL> INSERT INTO Admin\_List (Admin\_Num, Admin\_Address, Admin\_PhoneNum, Section\_Num, Admin\_LName, Admin\_FName, Admin\_ZipCode)

2 VALUES (AdminNum\_Seq.NEXTVAL,'504 Orange Park','111-333-3437', '4', 'Russell', 'Jim', '31004');

1 row inserted.

SQL> INSERT INTO Admin\_List (Admin\_Num, Admin\_Address, Admin\_PhoneNum, Section\_Num, Admin\_LName, Admin\_FName, Admin\_ZipCode)

2 VALUES (AdminNum\_Seq.NEXTVAL,'505 Orange Park','111-333-3438', '5', 'Bargueno', 'Patricia', '31004');

1 row inserted.

SQL> INSERT INTO Admin\_List (Admin\_Num, Admin\_Address, Admin\_PhoneNum, Section\_Num, Admin\_LName, Admin\_FName, Admin\_ZipCode)

2 VALUES (AdminNum\_Seq.NEXTVAL,'506 Orange Park','111-333-3439', '6', 'Hopkins', 'John', '31004');

1 row inserted.

SQL> INSERT INTO Admin\_List (Admin\_Num, Admin\_Address, Admin\_PhoneNum, Section\_Num, Admin\_LName, Admin\_FName, Admin\_ZipCode)

2 VALUES (AdminNum\_Seq.NEXTVAL,'507 Orange Park','111-333-3440', '7', 'Smith', 'Will', '31004');

1 row inserted.

SQL> INSERT INTO Admin\_List (Admin\_Num, Admin\_Address, Admin\_PhoneNum, Section\_Num, Admin\_LName, Admin\_FName, Admin\_ZipCode)

2 VALUES (AdminNum\_Seq.NEXTVAL,'508 Orange Park','111-333-3441', '8', 'Brown', 'Adam', '31004');

1 row inserted.

SQL> INSERT INTO Admin\_List (Admin\_Num, Admin\_Address, Admin\_PhoneNum, Section\_Num, Admin\_LName, Admin\_FName, Admin\_ZipCode)

2 VALUES (AdminNum\_Seq.NEXTVAL,'509 Orange Park','111-333-3442', '9', 'Williams', 'Shane', '31004');

1 row inserted.

SQL> INSERT INTO Admin\_List (Admin\_Num, Admin\_Address, Admin\_PhoneNum, Section\_Num, Admin\_LName, Admin\_FName, Admin\_ZipCode)

2 VALUES (AdminNum\_Seq.NEXTVAL,'510 Orange Park','111-333-3443', '10', 'Robins', 'Amber', '31004');

1 row inserted.

SQL> INSERT INTO Admin\_List (Admin\_Num, Admin\_Address, Admin\_PhoneNum, Section\_Num, Admin\_LName, Admin\_FName, Admin\_ZipCode)

2 VALUES (AdminNum\_Seq.NEXTVAL,'511 Orange Park','111-333-3444', '4', 'Hawkins', 'Tim', '31004');

1 row inserted.

SQL>

SQL> INSERT INTO Student\_List (Student\_Num, Student\_Address, Student\_PhoneNum, Student\_LName, Student\_FName, Student\_ZipCode)

2 VALUES (StudentNum\_Seq.NEXTVAL,'Dorm 1','222-333-3434', 'Gilbert', 'Sarah', '31005');

1 row inserted.

SQL> INSERT INTO Student\_List (Student\_Num, Student\_Address, Student\_PhoneNum, Student\_LName, Student\_FName, Student\_ZipCode)

2 VALUES (StudentNum\_Seq.NEXTVAL,'Dorm 2','222-333-3435', 'Myers', 'James', '31005');

1 row inserted.

SQL> INSERT INTO Student\_List (Student\_Num, Student\_Address, Student\_PhoneNum, Student\_LName, Student\_FName, Student\_ZipCode)

2 VALUES (StudentNum\_Seq.NEXTVAL,'Dorm 3','222-333-3436', 'Howard', 'Rico', '31005');

1 row inserted.

SQL> INSERT INTO Student\_List (Student\_Num, Student\_Address, Student\_PhoneNum, Student\_LName, Student\_FName, Student\_ZipCode)

2 VALUES (StudentNum\_Seq.NEXTVAL,'Dorm 4','222-333-3437', 'Bush', 'Donald', '31005');

1 row inserted.

SQL> INSERT INTO Student\_List (Student\_Num, Student\_Address, Student\_PhoneNum, Student\_LName, Student\_FName, Student\_ZipCode)

2 VALUES (StudentNum\_Seq.NEXTVAL,'Dorm 5','222-333-3438', 'Simmons', 'Bill', '31005');

1 row inserted.

SQL> INSERT INTO Student\_List (Student\_Num, Student\_Address, Student\_PhoneNum, Student\_LName, Student\_FName, Student\_ZipCode)

2 VALUES (StudentNum\_Seq.NEXTVAL,'Dorm 6','222-333-3439', 'Aultman', 'Richard', '31005');

1 row inserted.

SQL> INSERT INTO Student\_List (Student\_Num, Student\_Address, Student\_PhoneNum, Student\_LName, Student\_FName, Student\_ZipCode)

2 VALUES (StudentNum\_Seq.NEXTVAL,'Dorm 7','222-333-3440', 'Ruger', 'Victoria', '31005');

1 row inserted.

SQL> INSERT INTO Student\_List (Student\_Num, Student\_Address, Student\_PhoneNum, Student\_LName, Student\_FName, Student\_ZipCode)

2 VALUES (StudentNum\_Seq.NEXTVAL,'Dorm 8','222-333-3441', 'Thomas', 'Melissa', '31005');

1 row inserted.

SQL> INSERT INTO Student\_List (Student\_Num, Student\_Address, Student\_PhoneNum, Student\_LName, Student\_FName, Student\_ZipCode)

2 VALUES (StudentNum\_Seq.NEXTVAL,'Dorm 9','222-333-3442', 'Synder', 'Jennifer', '31005');

1 row inserted.

SQL> INSERT INTO Student\_List (Student\_Num, Student\_Address, Student\_PhoneNum, Student\_LName, Student\_FName, Student\_ZipCode)

2 VALUES (StudentNum\_Seq.NEXTVAL,'Dorm 10','222-333-3443', 'Baker', 'Elizabeth', '31005');

1 row inserted.

SQL> INSERT INTO Student\_List (Student\_Num, Student\_Address, Student\_PhoneNum, Student\_LName, Student\_FName, Student\_ZipCode)

2 VALUES (StudentNum\_Seq.NEXTVAL,'Dorm 11','222-333-3444', 'Pines', 'Matt', '31005');

1 row inserted.

SQL>

SQL>

SQL>

SQL> INSERT INTO Instr\_List (Instr\_Num, Instr\_Address, Instr\_PhoneNum, Section\_Num, Instr\_LName, Instr\_FName, Instr\_ZipCode)

2 VALUES (InstrNum\_Seq.NEXTVAL,'101 Apple Street','111-222-3434', '1', 'Roberts', 'John', '31003');

1 row inserted.

SQL> INSERT INTO Instr\_List (Instr\_Num, Instr\_Address, Instr\_PhoneNum, Section\_Num, Instr\_LName, Instr\_FName, Instr\_ZipCode)

2 VALUES (InstrNum\_Seq.NEXTVAL,'102 Apple Street','111-222-3435', '2', 'Russell', 'Brandon', '31003');

1 row inserted.

SQL> INSERT INTO Instr\_List (Instr\_Num, Instr\_Address, Instr\_PhoneNum, Section\_Num, Instr\_LName, Instr\_FName, Instr\_ZipCode)

2 VALUES (InstrNum\_Seq.NEXTVAL,'103 Apple Street','111-222-3436', '3', 'Lopez', 'Mike', '31003');

1 row inserted.

SQL> INSERT INTO Instr\_List (Instr\_Num, Instr\_Address, Instr\_PhoneNum, Section\_Num, Instr\_LName, Instr\_FName, Instr\_ZipCode)

2 VALUES (InstrNum\_Seq.NEXTVAL,'104 Apple Street','111-222-3437', '4', 'Monteor', 'Jim', '31003');

1 row inserted.

SQL> INSERT INTO Instr\_List (Instr\_Num, Instr\_Address, Instr\_PhoneNum, Section\_Num, Instr\_LName, Instr\_FName, Instr\_ZipCode)

2 VALUES (InstrNum\_Seq.NEXTVAL,'105 Apple Street','111-222-3438', '5', 'Hamm', 'Patricia', '31003');

1 row inserted.

SQL> INSERT INTO Instr\_List (Instr\_Num, Instr\_Address, Instr\_PhoneNum, Section\_Num, Instr\_LName, Instr\_FName, Instr\_ZipCode)

2 VALUES (InstrNum\_Seq.NEXTVAL,'106 Apple Street','111-222-3439', '6', 'Ingle', 'Tammy', '31003');

1 row inserted.

SQL> INSERT INTO Instr\_List (Instr\_Num, Instr\_Address, Instr\_PhoneNum, Section\_Num, Instr\_LName, Instr\_FName, Instr\_ZipCode)

2 VALUES (InstrNum\_Seq.NEXTVAL,'107 Apple Street','111-222-3440', '7', 'Jordan', 'Keith', '31003');

1 row inserted.

SQL> INSERT INTO Instr\_List (Instr\_Num, Instr\_Address, Instr\_PhoneNum, Section\_Num, Instr\_LName, Instr\_FName, Instr\_ZipCode)

2 VALUES (InstrNum\_Seq.NEXTVAL,'108 Apple Street','111-222-3441', '8', 'Cooper', 'Adam', '31003');

1 row inserted.

SQL> INSERT INTO Instr\_List (Instr\_Num, Instr\_Address, Instr\_PhoneNum, Section\_Num, Instr\_LName, Instr\_FName, Instr\_ZipCode)

2 VALUES (InstrNum\_Seq.NEXTVAL,'109 Apple Street','111-222-3442', '9', 'McCoy', 'Amber', '31003');

1 row inserted.

SQL> INSERT INTO Instr\_List (Instr\_Num, Instr\_Address, Instr\_PhoneNum, Section\_Num, Instr\_LName, Instr\_FName, Instr\_ZipCode)

2 VALUES (InstrNum\_Seq.NEXTVAL,'110 Apple Street','111-222-3443', '10', 'Fuller', 'Jana', '31003');

1 row inserted.

SQL> INSERT INTO Instr\_List (Instr\_Num, Instr\_Address, Instr\_PhoneNum, Section\_Num, Instr\_LName, Instr\_FName, Instr\_ZipCode)

2 VALUES (InstrNum\_Seq.NEXTVAL,'111 Apple Street','111-222-3444', '10', 'Xavier', 'Mark', '31003');

1 row inserted.

SQL>

SQL> INSERT INTO Course\_List (Course\_Num, Course\_Name, Admin\_Num , Course\_Desc, Course\_Hours, Section\_Num)

2 VALUES (CourseNum\_Seq.NEXTVAL, 'Intro to Computer Science',1, 'Intro to Computer Science','3', '1');

1 row inserted.

SQL> INSERT INTO Course\_List (Course\_Num, Course\_Name, Admin\_Num , Course\_Desc, Course\_Hours, Section\_Num)

2 VALUES (CourseNum\_Seq.NEXTVAL, 'Intro to Art',2, 'Intro to Art','3', '2');

1 row inserted.

SQL> INSERT INTO Course\_List (Course\_Num, Course\_Name, Admin\_Num , Course\_Desc, Course\_Hours, Section\_Num)

2 VALUES (CourseNum\_Seq.NEXTVAL, 'Intro to Aircraft mechanics',3, 'Intro to Aircraft mechanics','3', '3');

1 row inserted.

SQL> INSERT INTO Course\_List (Course\_Num, Course\_Name, Admin\_Num , Course\_Desc, Course\_Hours, Section\_Num)

2 VALUES (CourseNum\_Seq.NEXTVAL, 'Intro to Robotics',4, 'Intro to Robotics','3', '4');

1 row inserted.

SQL> INSERT INTO Course\_List (Course\_Num, Course\_Name, Admin\_Num , Course\_Desc, Course\_Hours, Section\_Num)

2 VALUES (CourseNum\_Seq.NEXTVAL, 'Intro to Databases',5, 'Intro to Databases','3', '5');

1 row inserted.

SQL> INSERT INTO Course\_List (Course\_Num, Course\_Name, Admin\_Num , Course\_Desc, Course\_Hours, Section\_Num)

2 VALUES (CourseNum\_Seq.NEXTVAL, 'Advanced English',6, 'Advanced English','3', '6');

1 row inserted.

SQL> INSERT INTO Course\_List (Course\_Num, Course\_Name, Admin\_Num , Course\_Desc, Course\_Hours, Section\_Num)

2 VALUES (CourseNum\_Seq.NEXTVAL, 'Intro to Physics',7, 'Intro to Physics','3', '7');

1 row inserted.

SQL> INSERT INTO Course\_List (Course\_Num, Course\_Name, Admin\_Num , Course\_Desc, Course\_Hours, Section\_Num)

2 VALUES (CourseNum\_Seq.NEXTVAL, 'Intro to Chemistry',8, 'Intro to Chemistry','3', '8');

1 row inserted.

SQL> INSERT INTO Course\_List (Course\_Num, Course\_Name, Admin\_Num , Course\_Desc, Course\_Hours, Section\_Num)

2 VALUES (CourseNum\_Seq.NEXTVAL, 'Intro to Spanish',9, 'Intro to Spanish','3', '9');

1 row inserted.

SQL> INSERT INTO Course\_List (Course\_Num, Course\_Name, Admin\_Num , Course\_Desc, Course\_Hours, Section\_Num)

2 VALUES (CourseNum\_Seq.NEXTVAL, 'Automotive mechanics Intro',10, 'Automotive mechanics Intro','3', '10');

1 row inserted.

SQL> INSERT INTO Course\_List (Course\_Num, Course\_Name, Admin\_Num , Course\_Desc, Course\_Hours, Section\_Num)

2 VALUES (CourseNum\_Seq.NEXTVAL, 'Advanced Robotics',11, 'Advanced Robotics','6', '4');

1 row inserted.

SQL>

SQL> INSERT INTO Class\_Sched (Sched\_Num, Sched\_Day, Sched\_Time, Sched\_Day\_Off, Sched\_Notes)

2 VALUES (SchedNum\_Seq.NEXTVAL,'Mon-Tues-Wed','0900-1100','Oct. 13th', 'This is the one of the main schedules');

1 row inserted.

SQL> INSERT INTO Class\_Sched (Sched\_Num, Sched\_Day, Sched\_Time, Sched\_Day\_Off, Sched\_Notes)

2 VALUES (SchedNum\_Seq.NEXTVAL,'Mon-Tues-Thur','0900-1100','Oct. 13th', 'This is the one of the main schedules');

1 row inserted.

SQL> INSERT INTO Class\_Sched (Sched\_Num, Sched\_Day, Sched\_Time, Sched\_Day\_Off, Sched\_Notes)

2 VALUES (SchedNum\_Seq.NEXTVAL,'Mon-Wed','1000-1200','Oct. 14th', 'This is the one of the main schedules');

1 row inserted.

SQL> INSERT INTO Class\_Sched (Sched\_Num, Sched\_Day, Sched\_Time, Sched\_Day\_Off, Sched\_Notes)

2 VALUES (SchedNum\_Seq.NEXTVAL,'Mon-Wed-Fri','1000-1200','Oct. 15th', 'This a alternative schedules');

1 row inserted.

SQL> INSERT INTO Class\_Sched (Sched\_Num, Sched\_Day, Sched\_Time, Sched\_Day\_Off, Sched\_Notes)

2 VALUES (SchedNum\_Seq.NEXTVAL,'Mon-Tues','0800-0900','Oct. 16th', 'This a alternative schedules');

1 row inserted.

SQL> INSERT INTO Class\_Sched (Sched\_Num, Sched\_Day, Sched\_Time, Sched\_Day\_Off, Sched\_Notes)

2 VALUES (SchedNum\_Seq.NEXTVAL,'Tues-Wed','1300-1500','Oct. 17th', 'This a alternative schedules');

1 row inserted.

SQL> INSERT INTO Class\_Sched (Sched\_Num, Sched\_Day, Sched\_Time, Sched\_Day\_Off, Sched\_Notes)

2 VALUES (SchedNum\_Seq.NEXTVAL,'Thur-Fri','1300-1500','Oct. 18th', 'This a alternative schedules');

1 row inserted.

SQL> INSERT INTO Class\_Sched (Sched\_Num, Sched\_Day, Sched\_Time, Sched\_Day\_Off, Sched\_Notes)

2 VALUES (SchedNum\_Seq.NEXTVAL,'Wed-Fri','1600-1800','Oct. 19th', 'This a alternative schedules');

1 row inserted.

SQL> INSERT INTO Class\_Sched (Sched\_Num, Sched\_Day, Sched\_Time, Sched\_Day\_Off, Sched\_Notes)

2 VALUES (SchedNum\_Seq.NEXTVAL,'Sat', '0900-1100','Oct. 20th', 'Weekend Schedule');

1 row inserted.

SQL> INSERT INTO Class\_Sched (Sched\_Num, Sched\_Day, Sched\_Time, Sched\_Day\_Off, Sched\_Notes)

2 VALUES (SchedNum\_Seq.NEXTVAL,'Sun', '0900-1100','Oct. 21st', 'Weekend Schedule');

1 row inserted.

SQL>

SQL> INSERT INTO Instr\_Classes (Instr\_Num, Sched\_Num, Course\_Num, Class\_Notes, Class\_Room)

2 VALUES ('1', '1', '1', 'Computer one broke','A');

1 row inserted.

SQL> INSERT INTO Instr\_Classes (Instr\_Num, Sched\_Num, Course\_Num, Class\_Notes, Class\_Room)

2 VALUES ('2', '2', '2', 'Room Ready','B');

1 row inserted.

SQL> INSERT INTO Instr\_Classes (Instr\_Num, Sched\_Num, Course\_Num, Class\_Notes, Class\_Room)

2 VALUES ('3', '3', '3', 'Instructor Chair broke','C');

1 row inserted.

SQL> INSERT INTO Instr\_Classes (Instr\_Num, Sched\_Num, Course\_Num, Class\_Notes, Class\_Room)

2 VALUES ('4', '4', '4', 'A/C not working','D');

1 row inserted.

SQL> INSERT INTO Instr\_Classes (Instr\_Num, Sched\_Num, Course\_Num, Class\_Notes, Class\_Room)

2 VALUES ('5', '5', '5', 'Computer two broke','E');

1 row inserted.

SQL> INSERT INTO Instr\_Classes (Instr\_Num, Sched\_Num, Course\_Num, Class\_Notes, Class\_Room)

2 VALUES ('6', '6', '6', 'No whiteboard','F');

1 row inserted.

SQL> INSERT INTO Instr\_Classes (Instr\_Num, Sched\_Num, Course\_Num, Class\_Notes, Class\_Room)

2 VALUES ('7', '7', '7', 'station 3 missing keyboard','G');

1 row inserted.

SQL> INSERT INTO Instr\_Classes (Instr\_Num, Sched\_Num, Course\_Num, Class\_Notes, Class\_Room)

2 VALUES ('8', '8', '8', 'Room Ready','H');

1 row inserted.

SQL> INSERT INTO Instr\_Classes (Instr\_Num, Sched\_Num, Course\_Num, Class\_Notes, Class\_Room)

2 VALUES ('9', '9', '9', 'Room Ready','I');

1 row inserted.

SQL> INSERT INTO Instr\_Classes (Instr\_Num, Sched\_Num, Course\_Num, Class\_Notes, Class\_Room)

2 VALUES ('10', '10', '10', 'Room Ready','J');

1 row inserted.

SQL> INSERT INTO Instr\_Classes (Instr\_Num, Sched\_Num, Course\_Num, Class\_Notes, Class\_Room)

2 VALUES ('10', '9', '10', 'Room Ready','J');

1 row inserted.

SQL> INSERT INTO Instr\_Classes (Instr\_Num, Sched\_Num, Course\_Num, Class\_Notes, Class\_Room)

2 VALUES ('9', '8', '10', 'Room Ready','X');

1 row inserted.

SQL> INSERT INTO Instr\_Classes (Instr\_Num, Sched\_Num, Course\_Num, Class\_Notes, Class\_Room)

2 VALUES ('1', '2', '9', 'Room Ready','Z');

1 row inserted.

SQL>

SQL>

SQL> INSERT INTO Student\_Class\_Signup (Student\_Num, Instr\_Num, Sched\_Num, Student\_Grade)

2 VALUES (1, 1, 1, 'A');

1 row inserted.

SQL> INSERT INTO Student\_Class\_Signup (Student\_Num, Instr\_Num, Sched\_Num, Student\_Grade)

2 VALUES (1, 2, 2, 'B');

1 row inserted.

SQL> INSERT INTO Student\_Class\_Signup (Student\_Num, Instr\_Num, Sched\_Num, Student\_Grade)

2 VALUES (2, 3, 3, 'A');

1 row inserted.

SQL> INSERT INTO Student\_Class\_Signup (Student\_Num, Instr\_Num, Sched\_Num, Student\_Grade)

2 VALUES (3, 3, 3, 'C');

1 row inserted.

SQL> INSERT INTO Student\_Class\_Signup (Student\_Num, Instr\_Num, Sched\_Num,Student\_Grade)

2 VALUES (4, 4, 4, 'C');

1 row inserted.

SQL> INSERT INTO Student\_Class\_Signup (Student\_Num, Instr\_Num, Sched\_Num, Student\_Grade)

2 VALUES (5, 5, 5, 'B');

1 row inserted.

SQL> INSERT INTO Student\_Class\_Signup (Student\_Num, Instr\_Num, Sched\_Num, Student\_Grade)

2 VALUES (6, 6, 6, 'D');

1 row inserted.

SQL> INSERT INTO Student\_Class\_Signup (Student\_Num, Instr\_Num, Sched\_Num, Student\_Grade)

2 VALUES (7, 7, 7, 'F');

1 row inserted.

SQL> INSERT INTO Student\_Class\_Signup (Student\_Num, Instr\_Num, Sched\_Num, Student\_Grade)

2 VALUES (8, 8, 8, '');

1 row inserted.

SQL> INSERT INTO Student\_Class\_Signup (Student\_Num, Instr\_Num, Sched\_Num, Student\_Grade)

2 VALUES (9, 9, 9, 'A');

1 row inserted.

SQL> INSERT INTO Student\_Class\_Signup (Student\_Num, Instr\_Num, Sched\_Num, Student\_Grade)

2 VALUES (10, 10, 10, 'A');

1 row inserted.

SQL>

SQL> commit;

Commit complete.

A class has been added!!

A class has been added!!

A class has been added!!

A class has been added!!

A class has been added!!

A class has been added!!

A class has been added!!

A class has been added!!

A class has been added!!

A class has been added!!

A class has been added!!

A class has been added!!

A class has been added!!

SQL>

SQL> /\* Verify that each table has 10 or more rows of data \*/

SQL>

SQL> SELECT /\*fixed\*/ \* FROM Section\_Info;

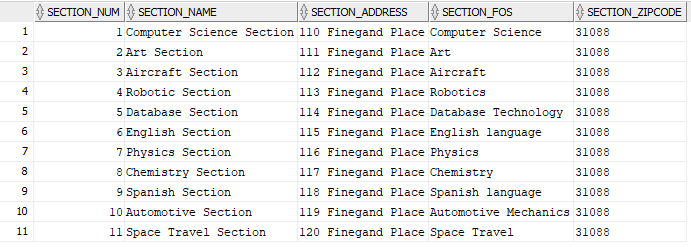


Figure 4. Graphical representation of Section Info SELECT.

11 rows selected.

SQL> SELECT /\*fixed\*/ \* FROM Instr\_List;

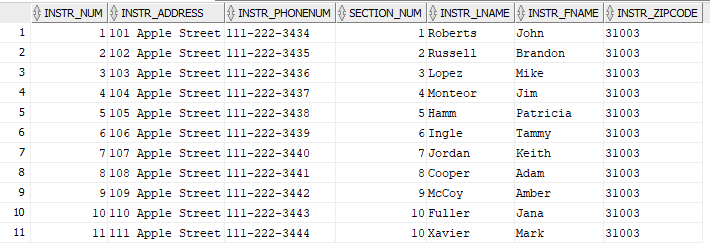


Figure 5. Graphical representation of Instructor List SELECT.

11 rows selected.

SQL> SELECT /\*fixed\*/ \* FROM Admin\_List;

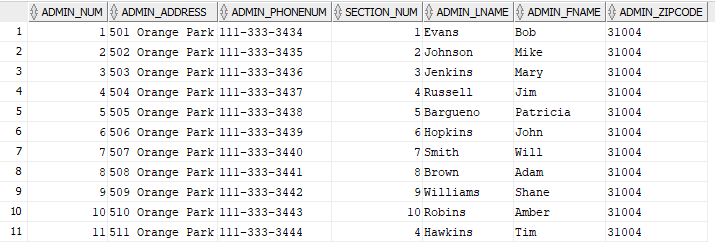


Figure 6. Graphical representation of Admin List SELECT.

11 rows selected.

SQL> SELECT /\*fixed\*/ \* FROM Student\_List;

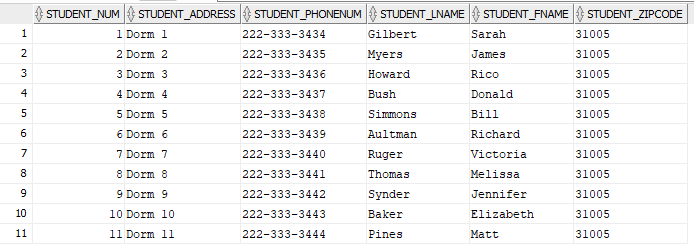


Figure 7. Graphical representation of Student List SELECT.

11 rows selected.

SQL> SELECT /\*fixed\*/ \* FROM Course\_List;

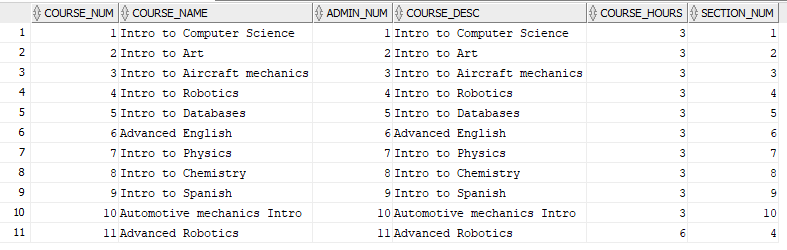


Figure 8. Graphical representation of Course List SELECT.

11 rows selected.

SQL> SELECT /\*fixed\*/ \* FROM Class\_Sched;

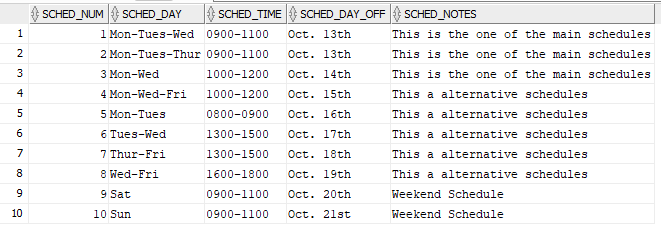


Figure 9. Graphical representation of Class Schedule SELECT.

10 rows selected.

SQL> SELECT /\*fixed\*/ \* FROM Instr\_Classes;

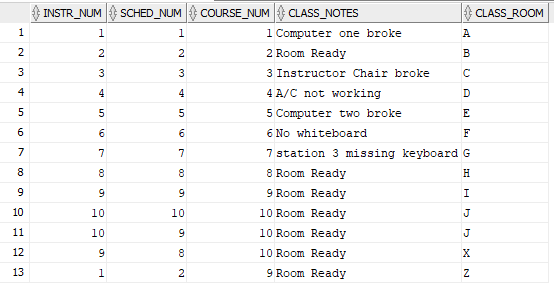


Figure 10. Graphical representation of Instructor Classes SELECT.

13 rows selected.

SQL> SELECT /\*fixed\*/ \* FROM Student\_Class\_Signup;

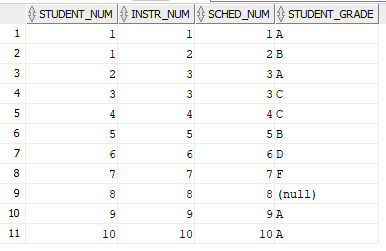


Figure 11. Graphical representation of Student Class Signup SELECT.

11 rows selected.

# Security

## Policies

Security policies are a subset of the security plan, defined in a previous section, and provide more detailed specification of what requirements will be enforced to promote database security hardening (Theriault & Heney, 1998). The following policies will be implemented in the Instructor Teaching Tracker database:

Username Policy. Effective immediately, all database users will be assigned a username. The username will consist of the first letter of the first name, followed by the full last name with no space, followed by the row identification number with no space.

Password Policy. Per the recommendation of (Grassi, et al., 2017), user passwords will be at least 8 characters in length, using a combination of all available Unicode characters, and should avoid common words or repetitive characters.

Connection Policy. All users are allowed to create a session to the database.

Role Assignment Policy. The admin\_role will be assigned to all administrative support staff. The instructor\_role will be assigned to all instructors. The student\_role will be assigned to all students.

Account Modify Policy. Administrators are the only authorized users to add or remove users from the instructor and student tables. Administrators can view and update their personal information in the admin table.

Course Management Policy. Administrators are in charge of adding, removing, and updating course table data.

Section Management Policy. Administrators are in charge of adding, removing, and updating section table data.

Class Schedule Modify Policy. Administrators are the only authorized user to add, remove, and update the class schedule table data. However, instructors can update the notes section.

Admin Support Policy. Administrators are authorized to insert, update, and delete all database tables, in the event other users need assistance. For example, instructors rely on administrators to add and remove classes from their schedule, and students would need an administrator to disenroll from a class.

Admin Account Restriction Policy. Administrators can view the admin table and update their personal information. The system administrators are the only ones able to add or remove from the admin table.

Instructor View Policy. Instructors are authorized to view only their personal info in the instructor list table. They may also view their section info, their class schedule, course information on all courses, and a complete list of available schedules to assign their classes to. They may also the name view basic information on students enrolled in their classes.

Instructor Modify Policy. Instructors are only authorized to update their personal information, their class notes, student grades of students enrolled in their classes, and may also remove students from their classes.

Student View Policy. Students are only authorized to view their personal information, the instructor class schedule, classes they are enrolled in, and their grade for classes they are enrolled in.

Student Modify Policy. Students are only authorized to update their personal information, and they may also enroll themselves in classes.

Class Tentative Schedule Policy. Only the Instructor DBA and school administrators may view and/or create tentative class schedules, i.e. alpha and beta test schedules. Tentative schedules will be regarded as highly sensitive data, while approved schedules will be regarded as sensitive data. Instructors and students may view approved class schedules.

## Procedures

Username Procedure. The following shows the SQL implementation procedure of the corresponding policy:

Password Procedure. The following shows the SQL implementation procedure of the corresponding policy:

Connection Procedure. The following shows the SQL implementation procedure of the corresponding policy:

Role Assignment Procedure. The following shows the SQL implementation procedure of the corresponding policy:

Account Modify Procedure. The following shows the SQL implementation procedure of the corresponding policy:

Course Management Procedure. The following shows the SQL implementation procedure of the corresponding policy:

Section Management Procedure. The following shows the SQL implementation procedure of the corresponding policy:

Class Schedule Modify Procedure. The following shows the SQL implementation procedure of the corresponding policy:

Admin Support Procedure. The following shows the SQL implementation procedure of the corresponding policy:

Admin Account Restriction Procedure. The following shows the SQL implementation procedure of the corresponding policy:

Instructor View Procedure. The following shows the SQL implementation procedure of the corresponding policy:

Instructor Modify Procedure. The following shows the SQL implementation procedure of the corresponding policy:

Student View Procedure. The following shows the SQL implementation procedure of the corresponding policy:

Student Modify Procedure. The following shows the SQL implementation procedure of the corresponding policy:

Class Tentative Schedule Policy. The following shows the SQL implementation procedure of the corresponding policy:

# Conclusion

conducted

## Lesson Learned

This

## Final Thoughts

The

References

Grassi, P. A., Fenton, J. L., Newton, E. M., Perlner, R. A., Regenscheid, A. R., Burr, W. E., . . . Theofanos, M. F. (2017). *NIST Special Publication 800-63B.* Gaithersburg: National Institute of Standards and Technology. doi:https://doi.org/10.6028/NIST.SP.800-63b

Theriault, M., & Heney, W. (1998). *Oracle Security.* O'Reilly Media, Inc.