**### Below please find the assignments for week 2:**

**# PRACTICE**

**## Foundations**

**## DFo\_4\_1\_sg**

Exercise 0: Installing Oracle SQL Developer Data Modeler

I was able to download the Oracle SQL Developer Data Modeler for MacOS.

Exercise 1: Identify and Create Entities, Attributes and Relationships

− Entities for a School Management System

- Parent\_information

- Student

- Student\_attendance

- Academic\_session

- Course

- Student\_course\_detail

- Department

- Exam\_result

- Online

- Seated

- Exam

- Exam\_type

- Faculty

- Faculty\_course\_detail

- Faculty\_login\_detail

- Attributes for each of the entities identified:

- parent\_information\_ID

- parent\_one\_first\_name

- parent\_one\_last\_name

- parent\_two\_first\_name

- parent\_two\_last\_name

- student\_ID

- student\_first\_name

- student\_last\_name

- registration\_year

- student\_email

- number\_of\_working\_days\_off

- number\_of\_days\_off

- exam\_eligibility

- academic\_session\_ID

- academic\_session\_name

- exam\_grade

- exam\_ID

- exam\_start\_date

- seated\_exam\_building

- seated\_exam\_room

- seated\_exam\_date

- seated\_exam\_time

- online\_logon\_ID

- online\_password

- course\_ID

- course\_name

- faculty\_contact\_hours

- faculty\_ID

- faculty\_first\_name

- faculty\_last\_name

- faculty\_email

- faculty\_full\_time\_salary

- faculty\_full\_time\_insurance\_plan

- faculty\_part\_time\_hourly\_rate

- department\_ID

- department\_head

- department\_name

- student\_course\_grade

− Relationship between the entities

- The relationships between the entities help represent the data connections within the database.

**## DFo\_4\_2\_sg**

Exercise 1: Forward Engineer a Logical Model to a Relational Model

**## DFo\_5\_1\_sg**

Exercise 1: Creating a Glossary from the Logical Model

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated

- Steps in order to complete this process include right clicking the design model in the browser and choosing the "properties" option. Then, "settings" must be expanded and the "naming standard" node has to be clicked. The plus icon needs the be click on which will lead to navigation to the glossary. For the naming standards, the "Engineer" icon needs to be pressed and once the Engineering to Relational Model dialog box presents, the "General Options" tab can be clicked on. The "Apply name translation" box should be marked. Click "Engineer."

**## DFo\_5\_2\_sg**

Exercise 1: Observe the mapping of the unique identifiers and relationship in the Relational Model

- Primary keys (PK) , unique keys (UK), foreign keys (FK). PK is a unique identifier, columns that identify. UK is a column that identifies uniqueness within the column. FKs allow for duplicates while PKs do not.

Exercise 2: Define table name abbreviations in csv file

- Abbreviations were made for each of the names in the plural table.

A screenshot of a computer

Description automatically generated

Exercise 3: Define Name Template

- In order to set the template, navigate to the Academic Database, right click, and choose "Properties." Adjust so table constraints are as follows:

A table with text on it

Description automatically generated

Exercise 4: Apply Name Template to the Relational Model

- In order to apply the name template to the relational model, the "Name Abbreviations" must be accessed by clicking on "Tools". Find the .csv file with the abbreviations and un-check "tables".

Exercise 5: Select how subtypes are generated in the Relational Model

- To select how the subtypes are generated in the Relational Model, the "Logical" tab has to be accessed. Access the "Faculty Super" in order to make edits. Choose "Subtypes" after clicking on "Options". Click on single-table from the Subtree Generation drop down menu, then re-engineer to the Relational Model.

**## DFo\_6\_1\_sg**

**Exercise 1: Introduction to Oracle Application Express**

* The guide to learn about the features within Oracle Application Express was reviewed. Oracle APEX allows for the creation of applications on databases. Also, it has features inclusive of interface themes, navigational controls, form handler, and flexible reports.

**## DFo\_6\_2\_sg**

**Exercise 1: Using Help in Oracle Application Express**

* To use Oracle Application Express effectively, practice is needed. Access the “help” icon which explores different facets such as Application Express SQL Workshop, Managing Database Objections with Object Browser, Using SQL Commands, and Using SQL Scripts. For example, within the “Object Browser,” it is possible to create various objects like tables and views.

**## DFo\_6\_3\_sg**

Exercise 1: Creating Tables Using Oracle Application Express

A diagram of a course

Description automatically generated

CREATE TABLE parent\_info (

id VARCHAR2(10) NOT NULL,

first\_name \_parent1 CHAR(50) NOT NULL,

last\_name \_parent1 CHAR(50) NOT NULL,

first\_name \_parent2 CHAR(50) NOT NULL,

last\_name \_parent2 CHAR(50) NOT NULL

);

CREATE TABLE student (

id VARCHAR2(10) NOT NULL,

first\_name CHAR(50) NOT NULL,

last\_name CHAR(50) NOT NULL,

registration\_yr NUMBER(4) NOT NULL,

email VARCHAR2(100) NOT NULL

);

CREATE TABLE student\_attendance (

nmbr\_working\_days INT NOT NULL,

nmbr\_days\_off INT NOT NULL,

exam\_elgibility VARCHAR2(50)

);

CREATE TABLE course (

id VARCHAR2(10) NOT NULL,

name VARCHAR2(50) NOT NULL

);

CREATE TABLE online (

id VARCHAR2(10) NOT NULL,

password VARCHAR2(50) NOT NULL

);

CREATE TABLE seated (

building VARCHAR2(10) NOT NULL,

room VARCHAR2(10) NOT NULL,

date\_time TIMESTAMP NOT NULL

);

CREATE TABLE academic\_session (

id VARCHAR2(10) NOT NULL,

name VARCHAR2(50) NOT NULL

);

CREATE TABLE exam\_result (

grade INT NOT NULL

);

CREATE TABLE exam (

id VARCHAR2(10) NOT NULL,

start\_date DATE

);

CREATE TABLE exam\_type (

id VARCHAR2(10) NOT NULL,

exam\_type VARCHAR2(50) NOT NULL,

name VARCHAR2(50) NOT NULL,

description VARCHAR2(1000)

);

CREATE TABLE department (

dept\_id VARCHAR2(10) NOT NULL,

name VARCHAR2(50) NOT NULL,

dept\_head CHAR(50)

);

CREATE TABLE faculty (

id VARCHAR2(10) NOT NULL,

first\_name CHAR(50) NOT NULL,

last\_name CHAR(50) NOT NULL,

email VARCHAR2(100) NOT NULL

);

CREATE TABLE faculty\_ft (

salary INT NOT NULL,

ins\_plan VARCHAR2(50) NOT NULL

);

CREATE TABLE faculty\_pt (

hourly\_wage INT NOT NULL

);

CREATE TABLE faculty\_course\_dtl (

contact\_hrs INT NOT NULL

);

CREATE TABLE faculty\_login\_dtl (

login\_date\_time TIMESTAMP NOT NULL

);

Exercise 2: Altering the Tables

The following fields should have unique values:  
• Course Name in AD\_COURSES  
• Department Name in AD\_DEPARTMENTS  
• Student Email in AD\_STUDENTS  
• Faculty Email in AD\_FACULTY  
• Session Name in AD\_ACADEMIC\_SESSIONS

CREATE TABLE parent\_info (

id VARCHAR2(10) NOT NULL,

first\_name \_parent1 CHAR(50) NOT NULL,

last\_name \_parent1 CHAR(50) NOT NULL,

first\_name \_parent2 CHAR(50) NOT NULL,

last\_name \_parent2 CHAR(50) NOT NULL,

student\_id VARCHAR2(10) NOT NULL,

CONSTRAINT parent\_id\_pk PRIMARY KEY (id),

CONSTRAINT student\_id\_fk FOREIGN KEY (student\_id) REFERENCES student (id)

);

CREATE TABLE student (

id VARCHAR2(10) NOT NULL,

First\_name CHAR(50) NOT NULL,

last\_name CHAR(50) NOT NULL,

resgistration\_yr NUMBER(4) NOT NULL,

email VARCHAR2(100) NOT NULL,

CONSTRAINT student\_id\_pk PRIMARY KEY (id),

CONSTRAINT parent\_id\_fk FOREIGN KEY (parent\_id)

REFERENCES parent\_info (id)

);

CREATE TABLE student\_attendance (

nmbr\_working\_days INT NOT NULL,

nmbr\_days\_off INT NOT NULL,

exam\_elgibility VARCHAR2(50),

CONSTRAINT student\_id\_uk, session\_id\_uk UNIQUE

student (id), academic\_session (id)

);

CREATE TABLE student\_course\_dtl (

grade INT NOT NULL,

CONSTRAINT student\_id\_uk, course\_id\_uk UNIQUE

student (id), course (id)

);

CREATE TABLE course (

id VARCHAR2(10) NOT NULL,

name VARCHAR2(50) NOT NULL,

CONSTRAINT course\_id\_pk PRIMARY KEY (id),

CONSTRAINT session\_id\_fk, online\_id\_fk, seated\_id\_fk, dept\_id\_fk FOREIGN KEY REFERENCES academic\_session (id), online (id), seated (id), department (id)

);

CREATE TABLE online (

logon\_id VARCHAR2(10) NOT NULL,

password VARCHAR2(50) NOT NULL,

CONSTRAINT logon\_id PRIMARY KEY (logon id),

CONSTRAINT course\_id\_fk KEY REFERENCES course (id)

);

CREATE TABLE seated (

building VARCHAR2(10) NOT NULL,

room VARCHAR2(10) NOT NULL,

date\_time TIMESTAMP NOT NULL,

CONSTRAINT building\_uk, room\_uk, date\_time\_uk UNIQUE (building, room, date\_time)

);

CREATE TABLE academic\_session (

id VARCHAR2(10) NOT NULL,

name VARCHAR2(50) NOT NULL,

CONSTRAINT session\_id\_pk PRIMARY KEY (id),

CONSTRAINT student\_id\_fk FOREIGN KEY REFERENCES student (id)

);

CREATE TABLE exam\_result (

grade INT NOT NULL,

CONSTRAINT student\_id\_uk, exam\_id\_uk, course\_id\_uk UNIQUE student (id), exam (id), course (id)

);

CREATE TABLE exam (

id VARCHAR2(10) NOT NULL,

start\_date DATE,

course\_id VARCHAR2(10) NOT NULL,

CONSTRAINT exam\_id\_pk PRIMARY KEY (id),

CONSTRAINT course\_id FOREIGN KEY REFERENCES course (id)

);

CREATE TABLE exam\_type (

type VARCHAR2(50) NOT NULL,

name VARCHAR2(50) NOT NULL,

description VARCHAR2(1000),

CONSTRAINT exam\_type\_pk PRIMARY KEY (type),

CONSTRAINT exam\_id\_fk FOREIGN KEY REFERENCES exam (id)

);

CREATE TABLE department (

id VARCHAR2(10) NOT NULL,

name VARCHAR2(50) NOT NULL,

dept\_head CHAR(50),

CONSTRAINT dept\_id\_pk PRIMARY KEY (id)

);

CREATE TABLE faculty (

id VARCHAR2(10) NOT NULL,

first\_name CHAR(50) NOT NULL,

last\_name CHAR(50) NOT NULL,

email VARCHAR2(100) NOT NULL,

full\_time\_id NUMBER(10),

part\_time\_id NUMBER(10),

CONSTRAINT faculty\_id\_pk PRIMARY KEY (id),

CONSTRAINT full\_time\_id\_fk, part\_time\_id\_fk FOREIGN KEY REFERENCES faculty\_ft (id), faculty\_pt (id)

);

CREATE TABLE faculty\_ft (

id VARCHAR2(10) NOT NULL,

salary INT NOT NULL,

ins\_plan VARCHAR2(50) NOT NULL,

CONSTRAINT full\_time\_id\_pk PRIMARY KEY (id),

CONSTRAINT faculty\_id\_fk FOREIGN KEY REFERENCES faculty (id)

);

CREATE TABLE faculty\_pt (

id VARCHAR2(10) NOT NULL,

hourly\_wage INT NOT NULL,

CONSTRAINT part\_time\_id\_pk PRIMARY KEY (id),

CONSTRAINT faculty\_id\_fk FOREIGN KEY REFERENCES faculty (id)

);

CREATE TABLE faculty\_course\_dtl (

id VARCHAR2(10) NOT NULL,

contact\_hrs INT NOT NULL,

faculty\_id VARCHAR2(10) NOT NULL,

course\_id VARCHAR2(10) NOT NULL,

CONSTRAINT faculty\_course\_id\_pk PRIMARY KEY (id),

CONSTRAINT faculty\_id\_fk, course\_id\_fk FOREIGN KEY REFERENCES faculty (id), course (id)

);

CREATE TABLE faculty\_login\_dtl (

login\_date\_time TIMESTAMP NOT NULL,

CONSTRAINT login\_date\_time\_pk PRIMARY KEY (login\_date\_time),

CONSTRAINT faculty\_id\_fk FOREIGN KEY REFERENCES faculty (id)

);

Exercise 3: Creating Composite Primary, Foreign and Unique Keys

A table with numbers and letters

Description automatically generated

CREATE TABLE dept (

dept\_id NUMBER(8),

dept\_name VARCHAR2(30),

loc\_id NUMBER(4),

CONSTRAINT dept\_id\_pk, loc\_id\_pk PRIMARY (dept\_id, loc\_id)

);

A close-up of a list

Description automatically generated

CREATE TABLE suppliers (

sup\_id NUMBER(15),

sup\_name VARCHAR2(30),

contact\_name NUMBER(4),

CONSTRAINT sup\_id\_uk, sup\_name\_uk PRIMARY (sup\_id, sup\_name)

);

**A screenshot of a computer

Description automatically generated**

CREATE TABLE products (

product\_id NUMBER(10),

sup\_id NUMBER(15),

sup\_name VARCHAR2(30),

CONSTRAINT product\_id\_pk PRIMARY KEY (product\_id),

CONSTRAINT sup\_id\_fk, sup\_name\_fk FOREIGN KEY REFERENCES suppliers (sup\_id, sup\_name)

);

**A table with numbers and letters

Description automatically generated**

CREATE TABLE dept\_sample (

dept\_id NUMBER(8),

dept\_name VARCHAR2(30),

loc\_id NUMBER(4),

CONSTRAINT dept\_id\_uk , dept\_name\_uk UNIQUE (dept\_id, dept\_name)

);

**## DFo\_6\_4\_sg**

Exercise 1: Inserting Rows in Tables

A blue and white striped box with black text

Description automatically generatedINSERT INTO AD\_ACADEMIC\_SESSIONS (ID, NAME)

VALUES (100, ‘SPRING SESSION’),

(200, ‘FALL SESSION’),

(300, ‘SUMMER SESSION’);

A blue and white box with text

Description automatically generated

INSERT INTO AD\_DEPARTMENTS (ID, NAME, HEAD)

VALUES (10, ‘ACCOUNTING’, ‘MARK\_SMITH’),

(20, ‘BIOLOGY’, ‘DAVE\_GOLD’),

(30, ‘COMPUTER\_SCIENCE’, ‘LINDA\_BROWN’),

(40, ‘SUMMER SESSION’, ‘ANITA\_TAYLOR’);

A table of names and numbers

Description automatically generated with medium confidence

INSERT INTO AD\_PARENT\_INFORMATION (PARENT1\_FN, PARENT1\_LN, PARENT2\_FN, PARENT2\_LN)

VALUES (600, ‘NEIL’, ‘SMITH’, ‘DORIS’, ‘SMITH’),

(610, ‘WILLIAM’, ‘BEN’, ‘NITA’, ‘BEN’),

(620, ‘SEAN’, ‘TAYLOR’, ‘RHEA’, ‘TAYLOR’),

(630, ‘DAVE’, ‘CARMEN’, ‘CATHY’, ‘CARMEN’),

(640, ‘JOHN’, ‘AUDRY’, ‘JANE’, ‘AUDRY’);

A table with numbers and names

Description automatically generated with medium confidence

INSERT INTO AD\_STUDENTS (FIRST\_NAME, LAST\_NAME, REG\_YEAR, EMAIL, PARENT\_ID)

VALUES (720, ‘JACK’, ‘SMITH’, ‘01-Jan-2012’,

‘JSMITH@SCHOOL.EDU’, ‘600’),

(730, ‘NOAH’, ‘AUDRY’, ‘01-Jan-2012’,

‘NAUDRY@SCHOOL.EDU’, ‘640’),

(740, ‘RHONDA’, ‘TAYLOR’, ‘01-Sep-2012’,

‘RTAYLOR@SCHOOL.EDU’, ‘620’),

(750, ‘ROBERT’, ‘BEN’, ‘01-Mar-2012’,

‘RBEN@SCHOOL.EDU’, ‘610’),

(760, ‘JEANNE’, ‘BEN’, ‘01-Mar-2012’,

‘JBEN@SCHOOL.EDU’, ‘610’),

(770, ‘MILLS’, ‘CARMEN’, ‘01-Apr-2013’,

‘MCARMEN@SCHOOL.EDU’, ‘630’);

A close-up of a table

Description automatically generated

INSERT INTO AD\_COURSES (ID, NAME, SESSION\_ID, DEPT\_ID, LOGON\_ID, PASSWORD, BUILDING, ROOM, DATE\_TIME)

VALUES (195, ‘CELL\_BIOLOGY’, 200, 20, NULL, NULL,

‘BUILDING\_D’, 401, ‘MWF\_9-10’ ),

(190, ‘PRINCIPLES\_OF\_ACCOUNTING’, 100, 10, NULL,

NULL, ‘BUILDING\_A’, 101, ‘MWF\_12-1’ ),

(191, ‘INTRODUCTION\_TO\_BUSINESS\_LAW’, 100, 10,

NULL, NULL, ‘BUILDING\_B’, 201, ‘THUR\_2-4’),

(192, ‘COST\_ACCOUNTING’, 100, 10, NULL, NULL,

‘BUILDING\_C’, 301, ‘TUES\_5-7’),

(193, ‘STRATEGIC\_TAX\_PLANNING\_FOR\_BUSINESS’,

100, 10, NULL, ‘TAX123’, ‘PASSWORD’, NULL, NULL,

NULL),

(194, ‘GENERAL\_BIOLOGY’, 200, 20, ‘BIO123’,

‘PASSWORD’, NULL, NULL, NULL);

A close-up of a list

Description automatically generated

INSERT INTO AD\_FACULTY (ID, FIRST\_NAME, LAST\_NAME, EMAIL, SALARY, INSURANCE, HOURLY\_RATE, DEPT\_ID)

VALUES (800, ‘JILL’, ‘MILLER’, ‘JMILL@SCHOOL.EDU’, 10000,

‘HEALTH’, NULL, 20),

(810, ‘JAMES’, ‘BORG’, ‘JBORG@SCHOOL.EDU’, 30000,

‘HEALTH,DENTAL’, NULL, 10),

(820, ‘LYNN’, ‘BROWN’, ‘LBROWN@SCHOOL.EDU’,

NULL, NULL, 50, 30),

(830, ‘ARTHUR’, ‘SMITH’, ‘ASMITH@SCHOOL.EDU’,

NULL, NULL, 40, 10),

(840, ‘SALLY’, ‘JONES’, ‘SJONES@SCHOOL.EDU’, 50000,

‘HEALTH,DENTAL,VISION’, NULL, 40);

A blue and white rectangular box with text

Description automatically generated

INSERT INTO AD\_EXAM\_TYPES (TYPE, NAME, DESCRIPTION)

VALUES (‘MCE’, ‘Multiple\_Choice\_Exams’,

‘CHOOSE\_MORE\_THAN\_ONE\_ANSWER’),

(‘TF’, ‘TRUE\_AND\_FALSE\_Exams’,

‘CHOOSE\_EITHER\_TRUE\_OR\_FALSE’),

(‘ESS’, ‘ESSAY\_Exams’, ‘WRITE\_PARAGRAPHS’),

(‘SA’, ‘SHORT\_ANSWER\_Exams’,

‘WRITE\_SHORT\_ANSWERS’),

(‘FIB’, ‘FILL\_IN\_THE\_BLANKS\_Exams’,

‘TYPE\_IN\_THE\_CORRECT\_ANSWER’)

A table with numbers and letters

Description automatically generated

INSERT INTO AD\_EXAMS (ID, START\_DATE, EXAM\_TYPE, COURSE\_ID)

VALUES (500, ‘12-Sep-2013’, ‘MCE’, 190),

(510, ‘15-Sep-2013’, ‘SA’, 191),

(520, ‘18-Sep-2013’, ‘FIB;, 192),

(530, ‘21- Mar-2014’, ‘ESS’, 193),

(540, ‘02-Apr-2014’, ‘TF’, 194);

A table with numbers and a number on it

Description automatically generated

INSERT INTO AD\_EXAMS\_RESULTS (STUDENT\_ID, COURSE\_ID, EXAM\_ID, EXAM\_GRADE)

VALUES (720, 190, 500, 91),

(730, 195, 540, 87),

(730, 194, 530, 85),

(750, 195, 510, 97),

(750, 191, 520, 78),

(760, 192, 510, 70),

(720, 193, 520, 97),

(750, 192, 500, 60),

(760, 192, 540, 65),

(760, 191, 530, 60);

**A blue and white calendar with black text

Description automatically generated**

INSERT INTO AD\_STUDENT\_ATTENDACE (STUDENT\_ID, SESSION\_ID, NUM\_WORK\_DAYS, NUM\_DAYS\_OFF, EXAM\_ELIGIBILITY)

VALUES (730, 200, 180, 11, ‘Y’),

(740, 300, 180, 12 ‘Y’),

(770, 300, 180, 13, ‘Y’),

(720, 100, 180, 21, ‘Y’),

(750, 100, 180, 14 ‘Y’),

(760, 200, 180, 15, ‘Y’);

**A table with numbers and letters

Description automatically generated**

INSERT INTO AD\_STUDENT\_COURSE\_DETAILS (STUDENT\_ID, COURSE\_ID, GRADE)

VALUES (720, 190, ‘A’)

(750, 192, ‘A’)

(760, 190, ‘B’)

(770, 194, ‘A’)

(720, 193, ‘B’)

(730, 191, ‘C’)

(740, 195, ‘F’)

(760, 192, ‘C’)

(770, 192, ‘D’)

(770, 193, ‘F’)

**A table with numbers and a list of information

Description automatically generated with medium confidence**

INSERT INTO AD\_FACULTY\_COURSE\_DETAILS (FACULTY\_ID, COURSE\_ID, CONTACT\_HRS)

VALUES (800, 192, 3)

(800, 193, 4)

(800, 190, 5)

(800, 191, 3)

(810, 194, 4)

(810, 195, 5)

**A close-up of a login

Description automatically generated**

INSERT INTO AD\_FACULTY\_LOGIN\_DETAILS (FACULTY\_ID, LOGIN\_DATE\_TIME)

VALUES (800, ‘01-JUN-17\_05.10.39.000000\_PM’),

(800, ‘01-JUN-17\_05.13.15.000000\_PM’),

(810, ‘01-JUN-17\_05.13.21.000000\_PM’),

(840, ‘01-JUN-17\_05.13.26.000000\_PM’),

(820, ‘01-JUN-17 05.13.31.000000 PM’),

(830, ‘01-JUN-17 05.13.36.000000 PM’);

Exercise 2: Updating Rows in the Tables

ALTER TABLE AD\_FACULTY\_LOGIN\_DETAILS

ADD DETAILS VARCHAR2(50);

**## DFo\_6\_5\_sg**

Exercise 1: Controlling Transactions

A close-up of a mail

Description automatically generated

The email field will remain after the Savepoint ALTER\_DONE.

A close-up of a computer code

Description automatically generated

**## DFo\_6\_6\_sg**

Exercise 1: Retrieving Columns from tables

Write a simple query to view the data inserted in the tables created for the academic database

SELECT

DEPARTMENT\_NAME

FROM

DEPARTMENT

Write a query to retrieve the exam grade obtained by each student for every exam attempted.

SELECT

STUDENT\_ID

EXAM\_RESULT

FROM

AD\_EXAM\_RESULT

Write a query to check if a student is eligible to take exams based on the number of days he/she attended classes

SELECT

STUDENT\_ID

FROM

AD\_STUDENT\_ATTENDANCE

Display the LOGIN\_DATE\_TIME for each faculty member.

SELECT

FACULTY\_ID

FACULTY\_LOGIN\_DATE\_TIME

FROM

FACULTY\_LOGIN\_DETAIL

Display the name of the Head of the Department for each of the Departments

SELECT

DEPARTMENT\_HEAD

FROM

AD\_DEPARTMENTS

Retrieve the student ID and first name for each student concatenated with literal text to look like this:  
720: FIRST NAME IS JACK

SELECT

STUDENT\_ID || ‘: FIRST NAME IS’ || FIRST\_NAME AS STUDENT\_INFORMATION

FROM

AD\_STUDENTS

Display all the distinct exam types from the AD\_EXAMS table.

SELECT

DISTINCT EXAM\_TYPE

FROM

AD\_EXAMS

**## DFo\_6\_7\_sg**

Exercise 1: Restricting Data Using SELECT

1. Display the course details for the Spring Session.

SELECT

FROM

AD\_COURSES

WHERE

SESSION\_ID

2. Display the details of the students who have scored more than 95.

SELECT

STUDENT ID

FROM

AD\_EXAM\_RESULT

WHERE

SCORE >95

3. Display the details of the students who have scored between 65 and 70.

SELECT

STUDENT\_ID

FROM

AD\_STUDENT\_EXAM\_RESULT

WHERE

SCORE >65 AND <70  
4. Display the students who registered after 01-Jun-2012.

SELECT

STUDENT\_ID

FROM

AD\_STUDENTS

WHERE

REG\_YEAR > ’01-JUN-2012’  
5. Display the course details for departments 10 and 30.

SELECT

DEPARTMENT\_ID

FROM

AD\_COURSES

WHERE

DEPARTMENT\_ID IN (10, 30)  
6. Display the details of students whose first name begins with the letter "J".

SELECT

STUDENT\_FIRST\_NAME

STUDENT\_LAST\_NAME

FROM

AD\_STUDENTS

WHERE

STUDENT\_FIRST\_NAME LIKE ‘J’  
7. Display the details of students who have opted for courses 190 or 193.

SELECT

STUDENT\_ID

COURSE\_ID

FROM

AD\_STUDENT\_COURSE\_DETAILS

WHERE

COURSE\_ID IN (190, 193)  
8. Display the course details offered by department 30 for the Fall Session (Session ID 200)

SELECT

AD\_COURSES

FROM

AD\_COURSES

WHERE

DEPARTMENT\_ID = 30 AND SESSION\_ID = 200   
9. Display the course details of courses not being offered in the summer and fall session (Session ID 200 and 300).

SELECT

COURSE\_DETAILS

FROM

AD\_COURSES

WHERE

SESSION\_ID NOT IN (200, 300)  
10. Display the course details for department 20.

SELECT

COURSE\_DETAILS

FROM

AD\_COURSES

WHERE

DEPARTMENT\_ID = 20

**## DFo\_6\_8\_sg**

Exercise 1: Sorting Data Using ORDER BY

Display all fields for each of the records in ascending order for the following tables:  
a. AD\_STUDENTS ordered by REG\_YEAR

SELECT

FROM

AD\_STUDENTS

ORDER BY

REG\_YEAR ASC

b. AD\_EXAM\_RESULTS ordered by STUDENT\_ID and COURSE\_ID

SELECT

FROM

AD\_EXAM\_RESULTS

ORDER BY

STUDENT\_ID ASC

COURSE\_ID ASC  
  
c. AD\_STUDENT\_ATTENDANCE ordered by STUDENT\_ID

SELECT

FROM

AD\_STUDENT\_ATTENDANCE

ORDER BY

STUDENT\_ID ASC  
  
d. AD\_DEPARTMENTS ordered by the department ID

SELECT

FROM

AD\_DEPARTMENTS

ORDER BY

DEPARTMENT\_ID ASC  
  
2. Display the percentage of days students have taken days off and sort the records based on the percentage calculated.

SELECT

STUDENT\_ID

NUMBER\_OF\_DAYS\_OFF

NUMBER\_OF\_WORKING\_DAYS\_OFF

(NUMBER\_OF\_DAYS\_OFF / NUMBER\_OF\_WORKING\_DAYS\_OFF) \* 100 ASDAYS\_OFF\_PERCENTAGE

FROM

AD\_STUDENT\_ATTENDANCE

ORDER BY

PERCENTAGE\_DAYS\_OFF DESC  
  
3. Display the top 5 students based on exam grade results.

SELECT

STUDENT\_ID

EXAM\_GRADE

FROM

AD\_EXAM\_RESULTS

ORDER BY

EXAM\_GRADE DESC

4. Display the parent details ordered by the parent ID.

SELECT

FROM

AD\_PARENT\_DETAILS

ORDER BY

PARENT\_ID ASC

**## DFo\_6\_9\_sg**

Exercise 1: Using JOINS in SQL Queries

Display the different courses offered by the departments in the school.

SELECT

COURSE\_NAME

DEPARTMENT\_NAME

FROM

AD\_COURSES

JOIN

AD\_DEPARTMENTS ONDEPARTMENT\_ID = DEPARTMENT\_ID

Display the courses offered in the Fall session.

SELECT

COURSE\_NAME

FROM

AD\_COURSES

WHERE

SESSION\_ID

What output would be generated when the given statement is executed?  
SELECT \* FROM AD\_EXAMS  
CROSS JOIN AD\_EXAM\_TYPES;

The rows for each would come together.