Alicia Bonavita

*Oracle Foundations 5,6*

**5-1 Mapping Entities and Attributes Practice**

Task 1 - Making a Glossary

* Using the Sport DDL file provided to us for this activity , I was then able to open the academic database. I imported the file using the file button and imported the document into the database. After importing the document I merged the file into a relational database format which provided me the tree structure for the working database file.
* After filling in the database framework I was able to build the glossary by going to the logical model button to the left hand side and right clicking. After right clicking I went through the options and clicked the create glossary file and started importing the attribute and and entities using the tree diagram. I specified key notes such as abbreviation to customer as cst. Or item\_list as itm\_list.

Task 2 - Engineering design to apply the glossary and naming standards

* We are going to right click the logical model and find the properties button
* After that we are going to find the naming standards and expand it by clicking the plus sign on the left and navigate to the glossary
* Then we are going to engineer this option into the relational database by clicking the ion that looks like the following (>>) we will click general options and then apply name translation check box and find the use preferred abbreviations
* After that then we will engineer the model using the preferred abbreviation we created in the glossary.

**5-2 Mapping Primary and Foreign Keys Practice**

Task 1

* If you take a look at the tree diagram , you will see 1 - 2 boxes underneath the attributes and you will see primary and foreign key constraints under each table it has a direct relationship with.
* So we are going to create a excel file that will map both unique identifier for both primary and unique keys and map them to their relationships with foreign keys
* We will do this by creating a table that will contain plural table name column , and a required abbreviation for each table

Task 2 - Defining abbreviations for key and constraints in .csv file

* Below you will see a template using the sports DDL details. We created a spread sheet application that listed plural names and corresponding abbreviations to define was is in the physical data model layer.

[oracle\_5\_2](https://docs.google.com/spreadsheets/d/1TtwfGHvUrKvDApZBkGkf4BcBlGgbEfx_7iBC-j7xfBI/edit?usp=sharing)

Task 3 - Define Name Template

* Now that we have a define template for keys and constraints for our sports ddl the next thing to do is to take the csv file and combining it with predefined variables
* In this care now we are going to do this within the academic database design by using the ‘Object browser and right clicking to find the properties tab.
* Using Task 3 table constraints provided we are going to set predefined variables as follows for our sports ddl file within the academic database.
* How we do this is by browsing the academic database and finding the object browser button on the right hand side. Once we have the object browser button in sight we will right click it and find the properties function. Which will lead us to setting - naming standards. We are going to open that up and find the option under the naming standard to find the template. We are then just going to copy and paste the table constraint into our sport ddl.

Task 4 - Applying template to the relational database

* This task is now just going to put our naming template into our logic . relational database model and apply it to our existing entities within the sports ddl.
* We are going to apply this template by clicking on the top bar to tool button and navigating to name abbreviations
* Then as previously done in task 2 we are going to take that same csv file we previously made
* Un-check the table and then click ok

Task 5 - Selecting how subtypes are generated in the relational model

* The next thing we are going to do is subtyping generation within our relational database model
* We will first need to define subtypes that we want to map by performing this within the logical tab
* We are going to click logical tab and double click faulty super type entity and we are going to go to the properties option
* From the properties menu we are going to select the subtype from the left hand panel
* It is then going drop a subtree generation drop down and we are going to select single table and click ok to confirm the changes were want for our subtypes
* After that we are going to re-engineer it by finding the button on top that looks like the following (<<) and then merge it into our sports ddl relational model.

**6-1 : Introduction to Oracle Application Express**

In this module were were introduced to Oracles Application Express

Task 1

* We are going to go through the learning paths for the course and access the APEX Learner guide. And then we are going to learn the features and Oracle application express
  + What is Apex?
    - Is a web application development, deployment, and maintenance tool that can be used in Oracle databases
    - This is a limited web browser programming environment
  + What features are in Apex?
    - Apex gives you build in features in areas such as
    - Interface themes, navigational controls, form handler, and flexible reports
  + What is Apex architecture?
    - No matter what type of Apex environment you use - development, application, ect.
    - It's built on URL requests and translated through Apex PL/SQL calls.
    - The application session is managed in database tables within Apex and uses requests in separate database sessions. Due to apex not using a dedicated database and relies on sessions it consumes less CPU resources.
  + Apex Environment?
    - The application is called a workspace, workspaces are private virtual database applications that allows for collaboration with multiple users to work within the same instance while keeping users objects , data, and applications individually.

**6- 2 Structured Query Language Practice**

Task 1 Using Oracle Application Express SQL workshop

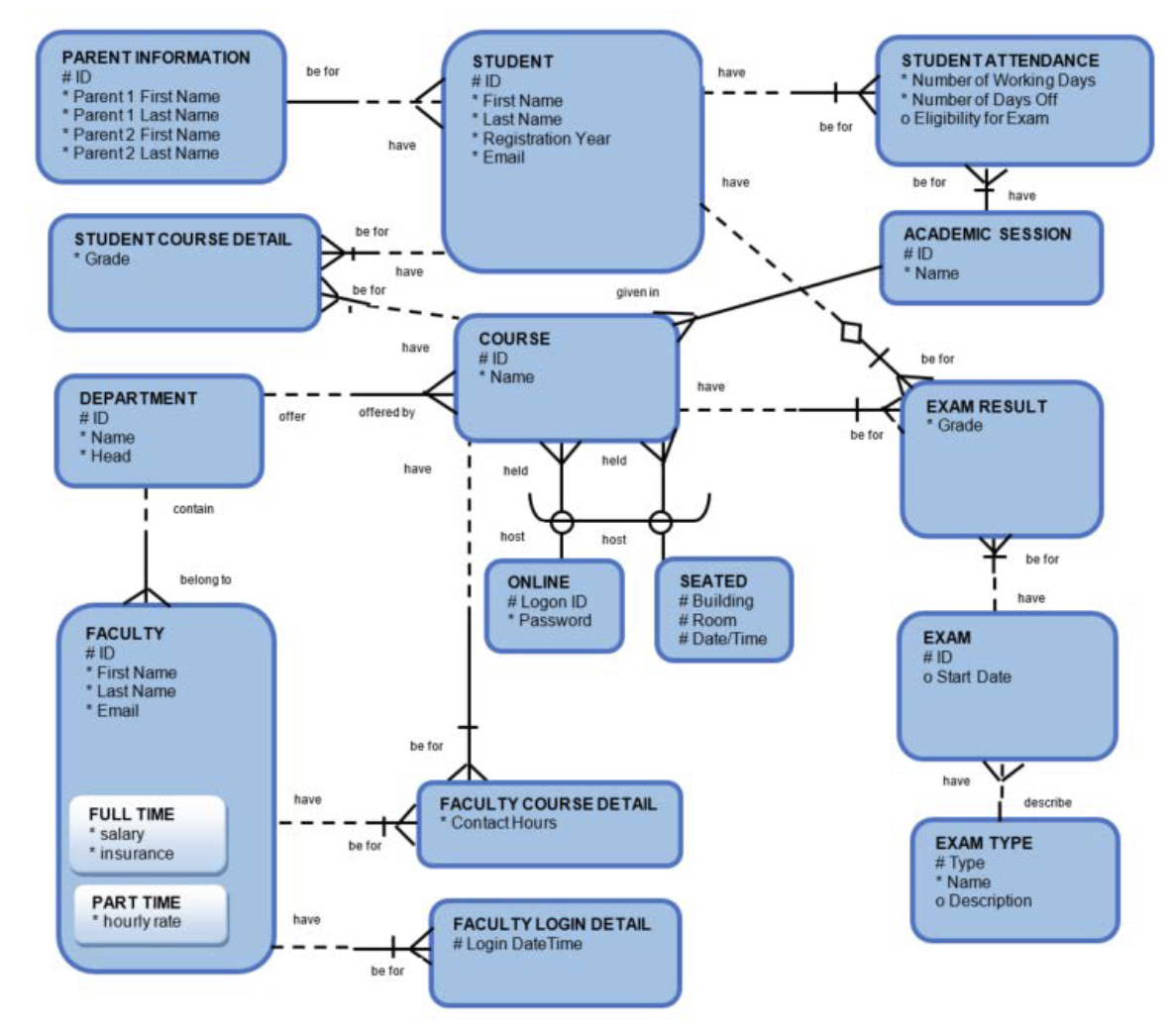
* In this task we are going to use oracle APEX and find the help icon and review

1. Managing database objects and object browser
   1. The Object Browser allows you to explore, create, and manage different database objects such as tables, views, indexes, sequences, etc.
   2. Use the tree view on the left to navigate through the different types of objects. Click on any object to view its details and available operations (like creating, editing, or deleting).
2. Using SQL commands
   1. This section provides a text editor where you can write and execute SQL queries.
   2. Write a SQL command in the editor and click the Run button to execute it. The results will be displayed in a table format below the editor.
3. Using SQL scripts
   1. This tool allows you to upload and manage multiple SQL scripts for batch execution.
   2. You can create a new script by clicking Upload or Create. After writing or uploading your script, click Run to execute it.

**Database Foundations 6-3: Defining Data Definition Language (DDL) Practices**

* Task 1: Creating Tables Using Oracle Application Express

Academic Database where the tables will be created



* + Create the DDL Statements for creating the tables for the Academic Database listed above – include NOT NULL constraints where necessary. (Other constraints will be added later)
    - 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,

resgistration\_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 student\_course\_dtl (

grade INT NOT NULL

);

* + - 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

);

* Task 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
  + Task 1: Alter the tables in the Academic Database to define the primary key, foreign key and unique constraints
    - 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),

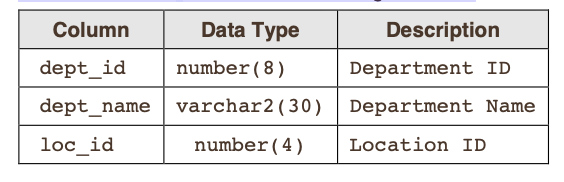
CONSTRAINT faculty\_id\_fk FOREIGN KEY REFERENCES faculty (id)

);

* + Task #2: Alter the table AD\_FACULTY\_LOGIN\_DETAILS and specify a default value for the column LOGIN\_DATE\_TIME of SYSDATE
    - ALTER TABLE AD\_FACULTY\_LOGIN\_DETAILS

MODIFY LOGIN\_DATE\_TIME SYSDATE NOT NULL

* + Task #3: Set the AD\_PARENT\_INFORMATION table to a read-only status
    - ALTER TABLE PARENT\_INFORMATION READ ONLY
* Task 3
* Creating Composite Primary, Foreign and Unique Keys
  + Task #1: The primary key for this table needs to be defined as a composite comprising of the dept\_id and loc\_id. Create the DEPT table with the following structure:



* + - CREATE TABLE dept (

dept\_id NUMBER(8),

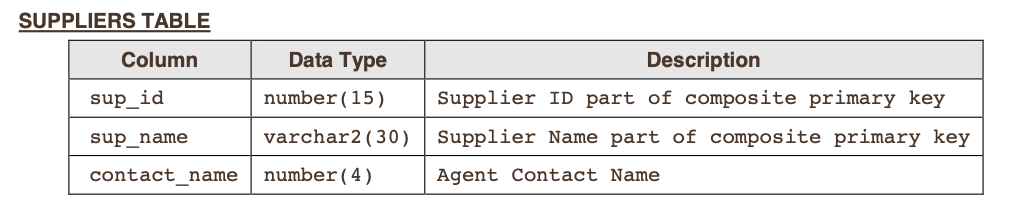
dept\_name VARCHAR2(30),

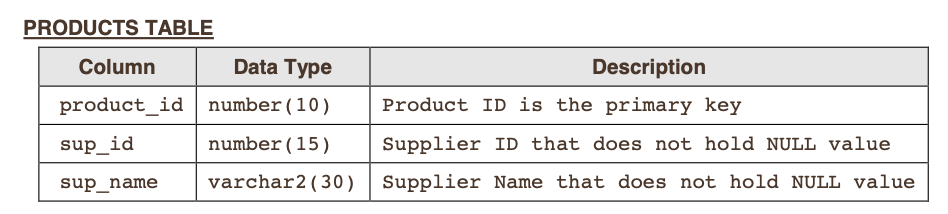
loc\_id NUMBER(4),

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

);

* + Task #2: The primary key for this table needs to be defined as a composite comprising of the sup\_id and sup\_name. The primary key for this table is product\_id. The foreign key for this table needs to be defined as a composite comprising of the sup\_id and sup\_name. Create the SUPPLIERS and PRODUCTS table with the following structure:





* + - 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)

);

* + - CREATE TABLE products (

product\_id NUMBER(10),

sup\_id NUMBER(15),

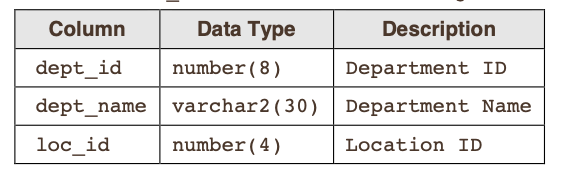
sup\_name VARCHAR2(30),

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

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

);

* + Task #3: The UNIQUE key for this table needs to be defined as a composite comprising of the dept\_id and dept\_name. Create the DEPT\_SAMPLE table with the following structure:



* + - CREATE TABLE dept\_sample (

dept\_id NUMBER(8),

dept\_name VARCHAR2(30),

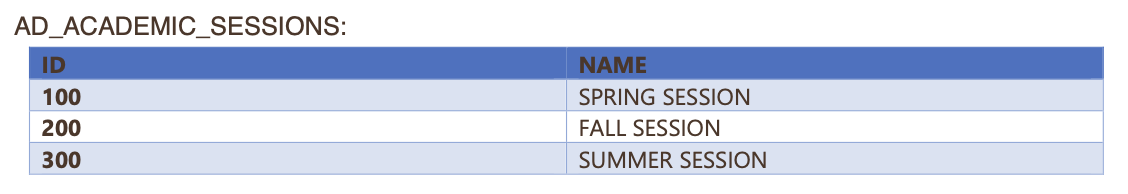
loc\_id NUMBER(4),

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

);

**Database Foundations 6-4: Defining Data Manipulation Practices**

* *Exercise 1:* Inserting Rows in Tables
  + Task #1: Insert rows into the tables created for the Academic Database based on the following tables

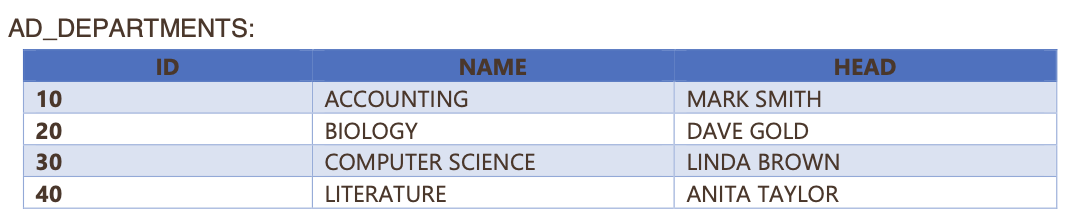


* + - INSERT INTO AD\_ACADEMIC\_SESSIONS (ID, NAME)

VALUES (100, ‘SPRING SESSION’),

(200, ‘FALL SESSION’),

(300, ‘SUMMER SESSION’);



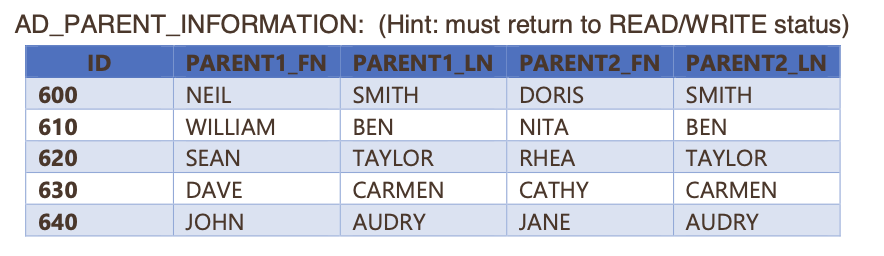
* + - INSERT INTO AD\_DEPARTMENTS (ID, NAME, HEAD)

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

(20, ‘BIOLOGY’, ‘DAVE\_GOLD’),

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

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



* + - 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’);



* + - 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’,

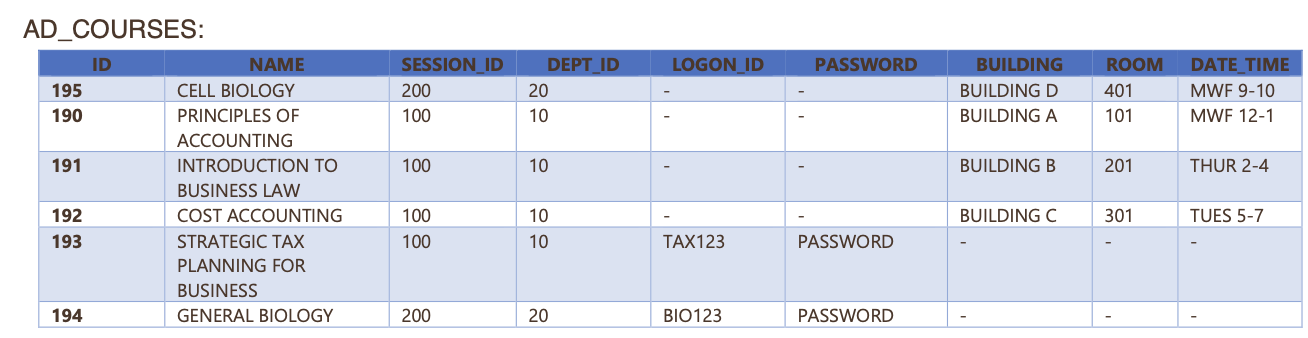
‘RBEN@SCHOOL.EDU’, ‘610’),

(760, ‘JEANNE’, ‘BEN’, ‘01-Mar-2012’,

‘JBEN@SCHOOL.EDU’, ‘610’),

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

‘MCARMEN@SCHOOL.EDU’, ‘630’);



* + - 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’),

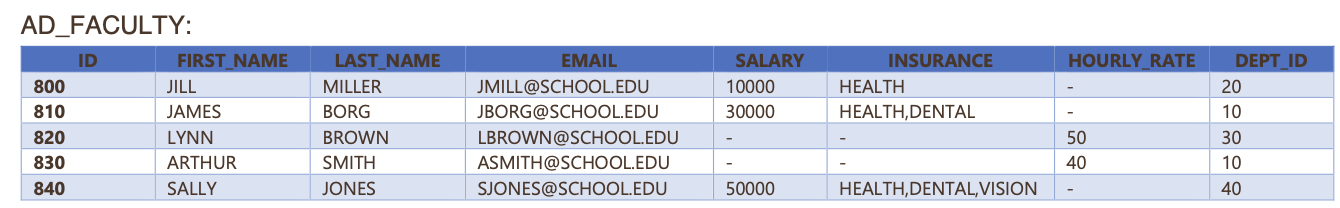
(193, ‘STRATEGIC\_TAX\_PLANNING\_FOR\_BUSINESS’,

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

NULL),

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

‘PASSWORD’, NULL, NULL, NULL);



* + - 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’,

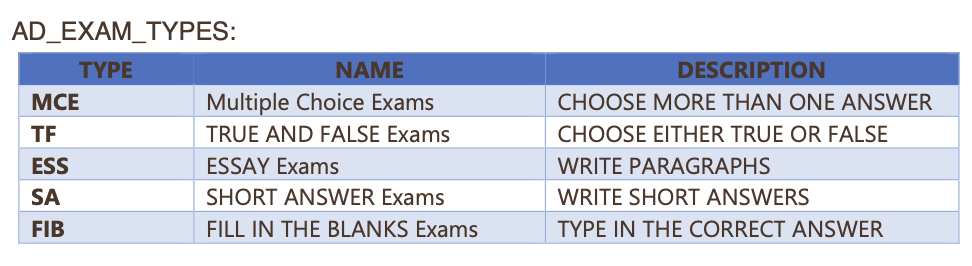
NULL, NULL, 50, 30),

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

NULL, NULL, 40, 10),

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

‘HEALTH,DENTAL,VISION’, NULL, 40);



* + - 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’),

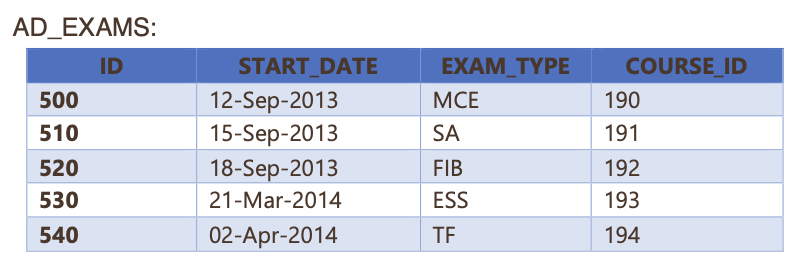
(‘ESS’, ‘ESSAY\_Exams’, ‘WRITE\_PARAGRAPHS’),

(‘SA’, ‘SHORT\_ANSWER\_Exams’,

‘WRITE\_SHORT\_ANSWERS’),

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

‘TYPE\_IN\_THE\_CORRECT\_ANSWER’)



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

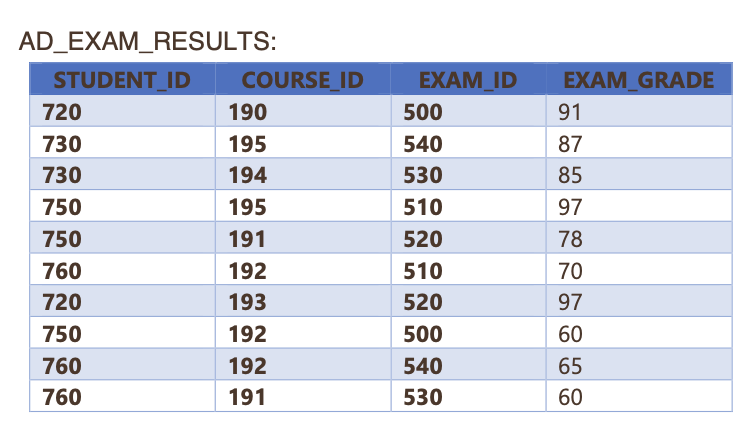
VALUES (500, ‘12-Sep-2013’, ‘MCE’, 190),

(510, ‘15-Sep-2013’, ‘SA’, 191),

(520, ‘18-Sep-2013’, ‘FIB;, 192),

(530, ‘21- Mar-2014’, ‘ESS’, 193),

(540, ‘02-Apr-2014’, ‘TF’, 194);



* + - 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),

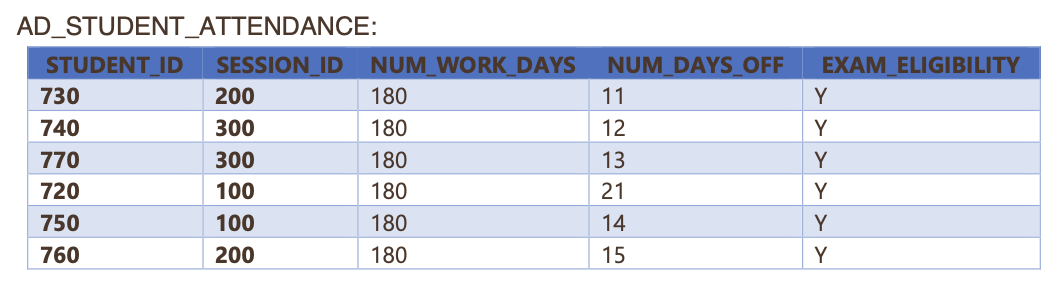
(760, 192, 510, 70),

(720, 193, 520, 97),

(750, 192, 500, 60),

(760, 192, 540, 65),

(760, 191, 530, 60);



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

VALUES (730, 200, 180, 11, ‘Y’),

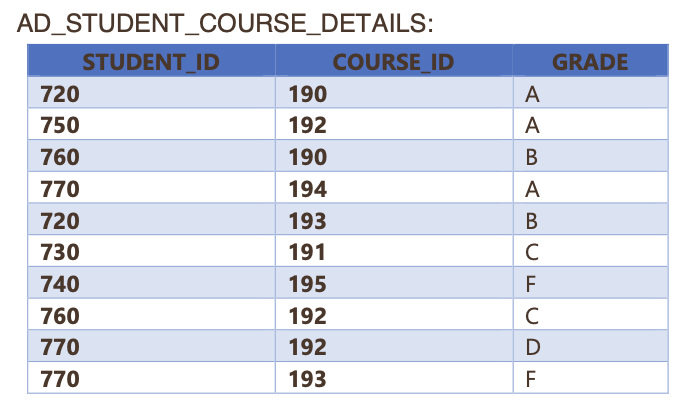
(740, 300, 180, 12 ‘Y’),

(770, 300, 180, 13, ‘Y’),

(720, 100, 180, 21, ‘Y’),

(750, 100, 180, 14 ‘Y’),

(760, 200, 180, 15, ‘Y’);



* + - 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’)

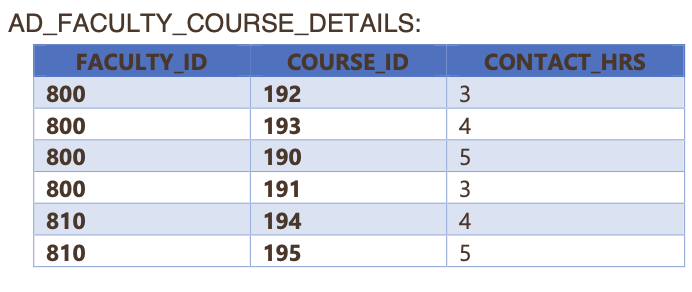
(730, 191, ‘C’)

(740, 195, ‘F’)

(760, 192, ‘C’)

(770, 192, ‘D’)

(770, 193, ‘F’)



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

VALUES (800, 192, 3)

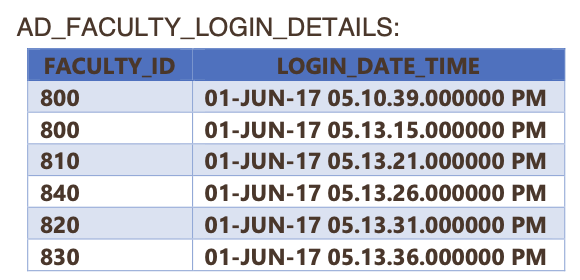
(800, 193, 4)

(800, 190, 5)

(800, 191, 3)

(810, 194, 4)

(810, 195, 5)



* + - 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’);

* 2: Updating Rows in the Tables
  + Task #1: Alter the AD\_FACULTY\_LOGIN\_DETAILS table to add a field called DETAILS make it a VARCHAR2(50) character field – it can have null values.
    - ALTER TABLE AD\_FACULTY\_LOGIN\_DETAILS

ADD DETAILS VARCHAR2(50);

* + Task #2: Update at least 2 records in the DETAILS column in the faculty login details table.
    - UPDATE AD\_FACULTY\_LOGIN\_DETAILS

SET DETAILS = ‘NOT\_UPDATED’

WHERE ID = 1;

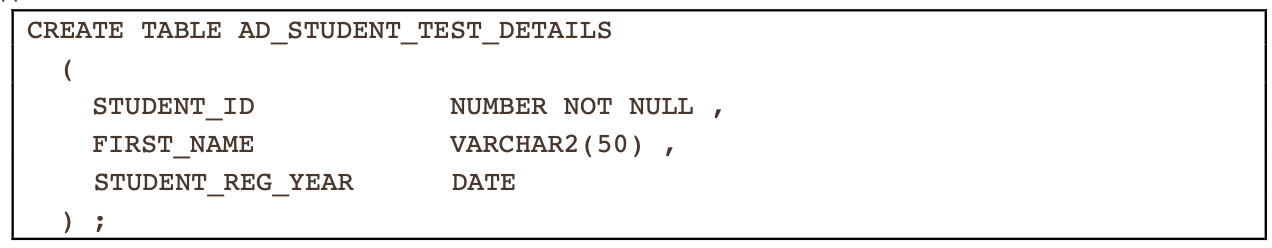
* + - UPDATE AD\_FACULTY\_LOGIN\_DETAILS

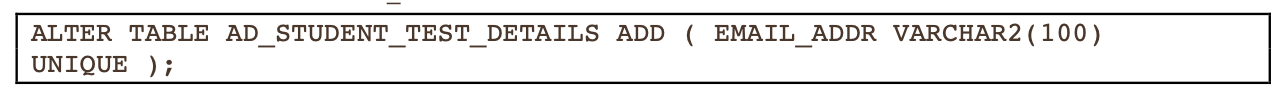
SET DETAILS = ‘UPDATED’

WHERE ID = 2;

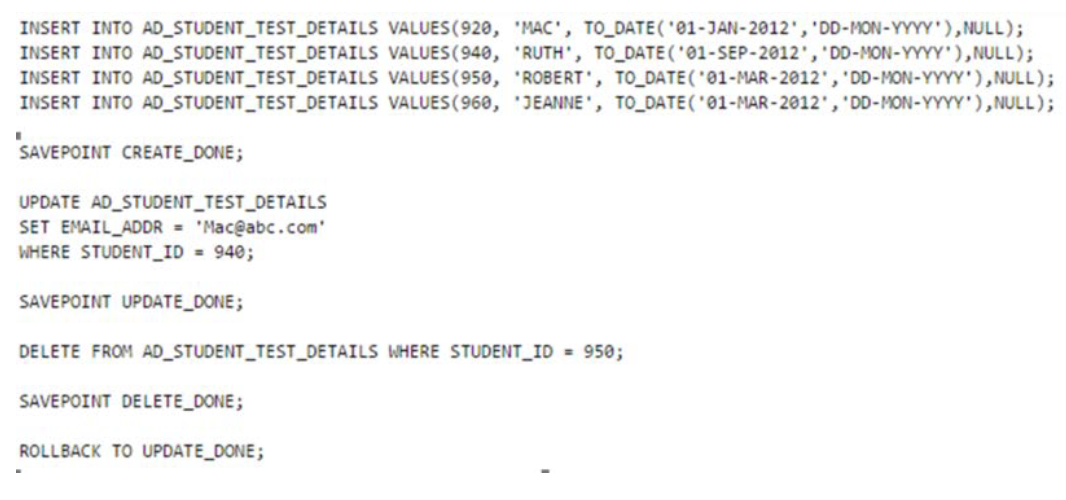
**Database Foundations 6-5: Defining Transaction Control Practices**

* Task 1: Controlling Transactions
  + Suppose a table with the following structure is created. Then the table is altered to add an email\_addr column. After the ALTER a Savepoint is created called ALTER\_DONE. A ROLLBACK is issued after the Savepoint ALTER\_DONE. Would the new email field still be there?





* + - The new email field will be there if the ROLLBACK specifies the Savepoint of ALTER\_DONE because the ALTER\_DONE Saverpoint includes the addition of the column
  + 2: If an INSERT is done to add rows into the test table and a Savepoint is then created called INSERT\_DONE. Then an UPDATE to a row in the test table is done and a Savepoint is created called UPDATE\_DONE. Then a DELETE is executed to delete a row in the test table and a Savepoint is created called DELETE\_DONE. At this point what records would be in the table? Then a ROLLBACK to Savepoint UPDATE\_DONE is issued. What changes would you notice with respect to the transactions and the records remaining in the table?



* + - For the first question, at that point, the records that are in the table after the DELETE\_DONE Savepoint are in the table. If a ROLLBACK to Savepoint UPDATE\_DONE is issued then the records remaining in the table are from before the DELETE was executed

**Database Foundations 6-6: Retrieving Data Practices**

* Task 1: Retrieving Columns from Tables
  + Task 1: Write a simple query to view the data inserted in the tables created for the academic database
    - For example, to view the data inserted into the parent information table:
      * SELECT \*

FROM AD\_PARENT\_INFORMATION;

* + 2: Write a query to retrieve the exam grade obtained by each student for every exam attempted
    - SELECT \*

FROM AD\_EXAMS\_RESULTS;

* + 3: Write a query to check if a student is eligible to take exams based on the number of days he/she attended classes
    - SELECT \*

FROM AD\_STUDENT\_ATTENDACE;

* + 4: Display the LOGIN\_DATE\_TIME for each faculty member
    - SELECT LOGIN\_DATE\_TIME

FROM AD\_FACULTY\_LOGIN\_DETAILS;

* + 5: Display the name of the Head of the Department for each of the Departments
    - SELECT HEAD

FROM AD\_DEPARTMENTS;

* + 6: 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;

* + 7: Display all the distinct exam types from the AD\_EXAMS table
    - SELECT DISTINCT TYPE

FROM AD\_EXAMS;

**6-7: Restricting Data Using WHERE Statement**

EXERCISE 1:

1. Display the course details for the Spring Session.

**SELECT** \*

**FROM** AD\_COURSES

**WHERE** SESSION\_ID = 100;

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

**SELECT** \*

**FROM** AD\_EXAM\_RESULTS

**WHERE** GRADE > 95;

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

**SELECT** \*

**FROM** AD\_EXAM\_RESULTS

**WHERE** GRADE **BETWEEN** 65 **AND** 70;

1. Display the students who registered after 01-June-2012.

**SELECT** \*

**FROM** AD\_STUDENTS

**WHERE** REG\_YEAR > ‘01-JUNE-2012’;

1. Display the course details for departments 10 and 30.

**SELECT** \*

**FROM** AD\_COURSES

**WHERE** DEPT\_ID **IN** (10, 30);

1. Display the details of students whose first name begins with the letter "J"

**SELECT** \*

**FROM** AD\_STUDENTS

**WHERE** FIRST\_NAME **LIKE** ‘J%’;

1. Display the details of students who have opted for courses 190 or 193.

**SELECT** \*

**FROM** AD\_STUDENT\_COURSE\_DETAILS

**WHERE** COURSE\_ID **IN** (190, 193);

1. Display the course details offered by department 30 for the Fall Session (Session ID 200).

**SELECT** \*

**FROM** AD\_COURSES

**WHERE** DEPT\_ID = 30 **AND** SESSION\_ID = 200;

1. Display the course details of courses not being offered in the summer and fall session (Session ID 200 and 300).

**SELECT \***

**FROM**  AD\_COURSES

**WHERE** SESSION\_ID **NOT IN** (200, 300);

1. Display the course details for department 20.

**SELECT** \*

**FROM** AD\_COURSES

**WHERE** DEPT\_ID = 20;

**6-8: Sorting Data Using ORDER BY Practices**

EXERCISE 1:

1. Display all fields for each of the records in ascending order for the following tables:

1. AD\_STUDENTS ordered by REG\_YEAR

**SELECT** \*

**FROM** AD\_STUDENTS

**ORDER BY** REG\_YEAR **ASC;**

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

**SELECT** \*

**FROM** AD\_EXAM\_RESULTS

**ORDER BY** STUDENT\_ID **ASC**, COURSE\_ID **ASC;**

1. AD\_STUDENT\_ATTENDANCE ordered by STUDENT\_ID

**SELECT** \*

**FROM** AD\_STUDENT\_ATTENDANCE

**ORDER BY** STUDENT\_ID **ASC;**

1. 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, (NUM\_DAYS\_OFF / NUM\_WORK\_DAYS) \* 100 **AS** ABSENCE\_PERCENTAGE

**FROM** AD\_STUDENT\_ATTENDANCE

**ORDER BY** ABSENCE\_PERCENTAGE **DESC;**

3. Display the top 5 students based on exam grade results.

**SELECT** STUDENT\_ID, GRADE

**FROM** AD\_EXAM\_RESULTS

**ORDER BY** GRADE **DESC**

**LIMIT** 5;

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

**SELECT** \*

**FROM** AD\_PARENTS

**ORDER BY** PARENT\_ID **ASC;**

**6-9: Joining Tables Using JOIN Practices**

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

**SELECT** C.COURSE\_NAME, D.DEPT\_NAME

**FROM** AD\_COURSES C

**JOIN** AD\_DEPARTMENTS D **ON** C.DEPARTMENT\_ID = D.DEPARTMENT\_ID;

1. Display the courses offered in the Fall session.

**SELECT** COURSE\_NAME

**FROM** AD\_COURSES

**WHERE** SESSION\_ID = 200;

1. Display the course details, the department that offers the courses and students who have enrolled for those courses.

**SELECT** C.COURSE\_NAME, D.DEPT\_NAME, S.STUDENT\_NAME

**FROM** AD\_COURSES C

**JOIN** AD\_DEPARTMENTS D **ON** C.DEPARTMENT\_ID = D.DEPARTMENT\_ID  
**JOIN** AD\_ENROLLMENTS E **ON** C.COURSE\_ID = E.COURSE\_ID  
**JOIN** AD\_STUDENTS S **ON** E.STUDENT\_ID = S.STUDENT\_ID;

1. Display the course details, the department that offers the courses and students who have enrolled for those courses for department 20.

**SELECT** C.COURSE\_NAME, D.DEPT\_NAME, S.STUDENT\_NAME

**FROM** AD\_COURSES C

**JOIN** AD\_DEPARTMENTS D **ON** C.DEPARTMENT\_ID = D.DEPARTMENT\_ID  
**JOIN** AD\_ENROLLMENTS E **ON** C.COURSE\_ID = E.COURSE\_ID  
**JOIN** AD\_STUDENTS S **ON** E.STUDENT\_ID = S.STUDENT\_ID

**WHERE** D.DEPARTMENT\_ID = 20;

1. Write a query to display the details of the exam grades obtained by students who have opted for the course with COURSE\_ID in the range of 190 to 192.

**SELECT** S.STUDENT\_NAME, .GRADE

**FROM** AD\_EXAM\_RESULTS E

**JOIN** AD\_STUDENTS S **ON** E.DEPARTMENT\_ID = S.STUDENT\_ID

**WHERE**  E.COURSE\_ID **BETWEEN** 190 **AND** 192;

1. Retrieve the rows from the AD\_EXAM\_RESULTS table even if there are no matching records in the AD\_COURSES table.

**SELECT** E.\*, C.COURSE\_NAME

**FROM** AD\_EXAM\_RESULTS E

**LEFT JOIN** AD\_COURSES C **ON** E.COURSE\_ID = C.COURSE\_ID

1. What output would be generated when the given statement is executed?

SELECT \* FROM AD\_EXAMS

CROSS JOIN AD\_EXAM\_TYPES;  
  
This would combine every row from the AD\_EXAMS with every row from AD\_EXAM\_TYPES.