ORACLE: Foundations 5, 6

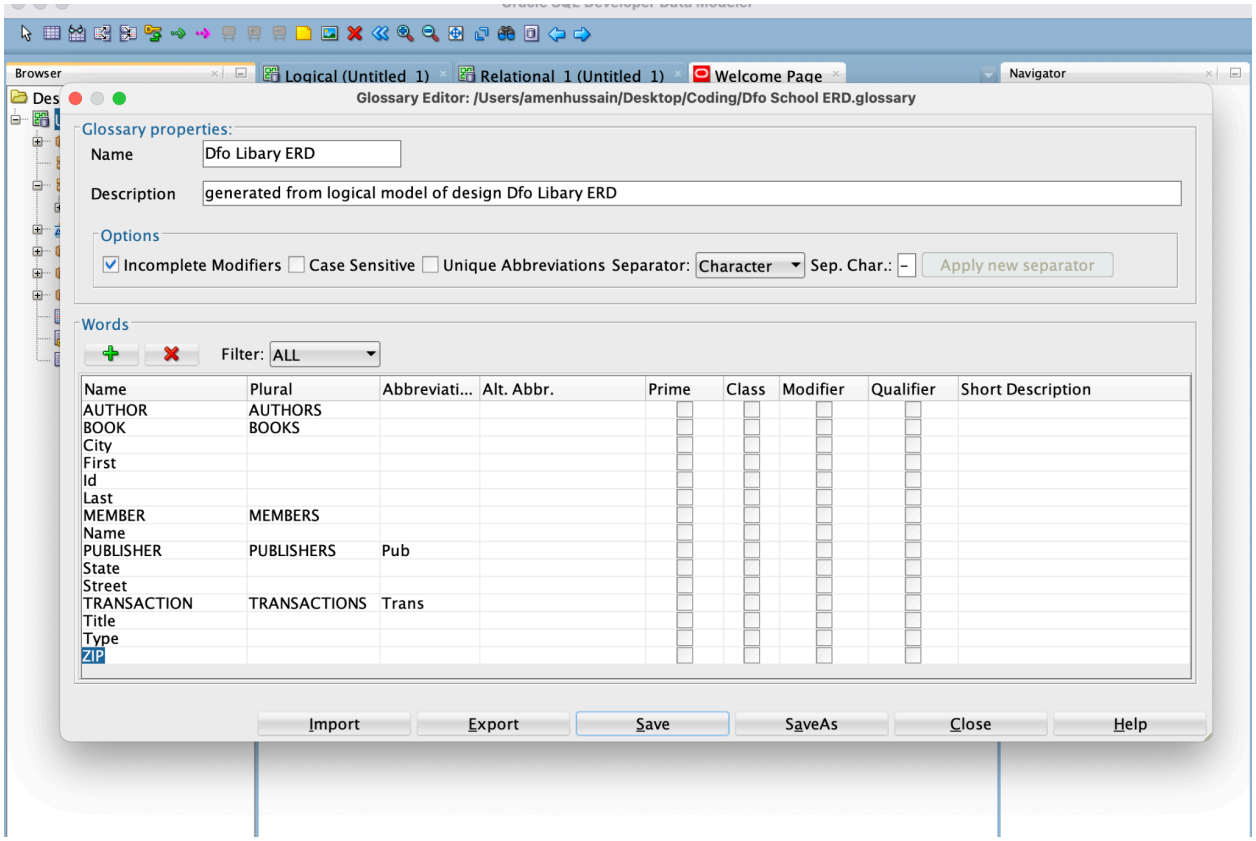
5-1: Mapping Entities and Attributes

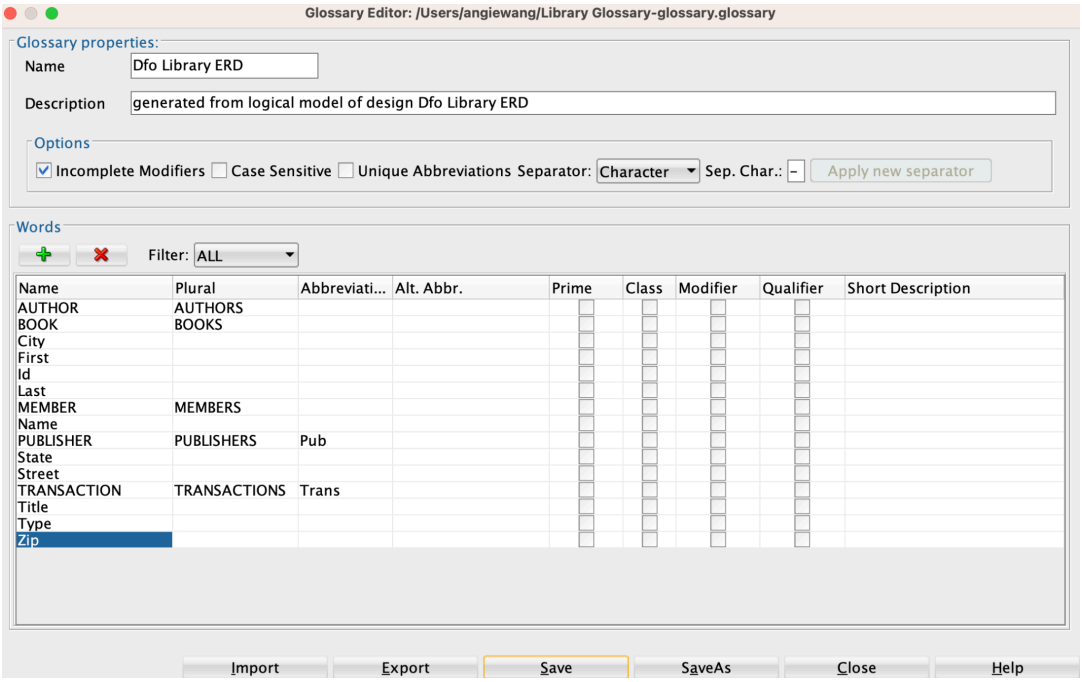
Exercise 1: Creating a Glossary from the Logical Model

Overview: In this practice, you create a Glossary from the Academic Database Logical Model.

Tasks

1. Open the Logical Model of the Academic Database
2. Right click the Logical Model node in the Browser and select "Create Glossary from Logical Model"
3. Specify the name of the Glossary, a brief description and specify as many classification types as applicable to the glossary entry.
4. Save the Glossary



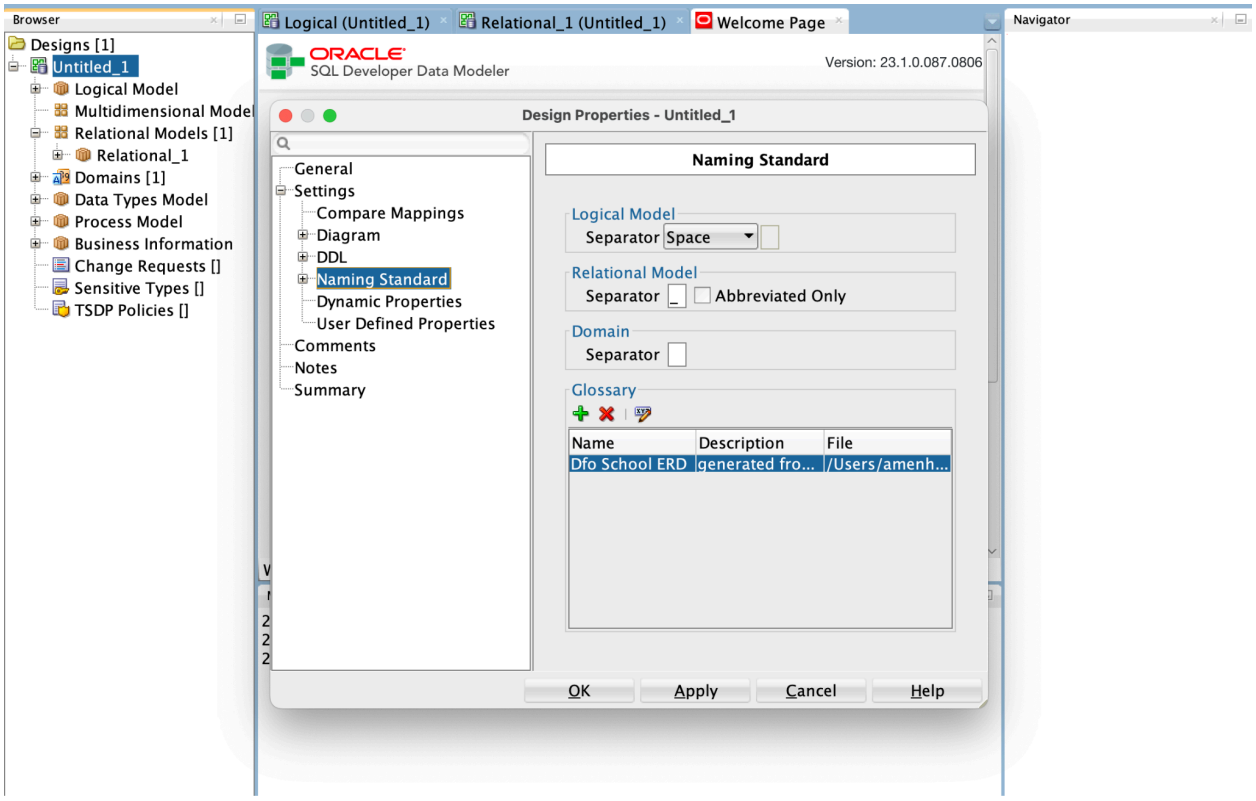


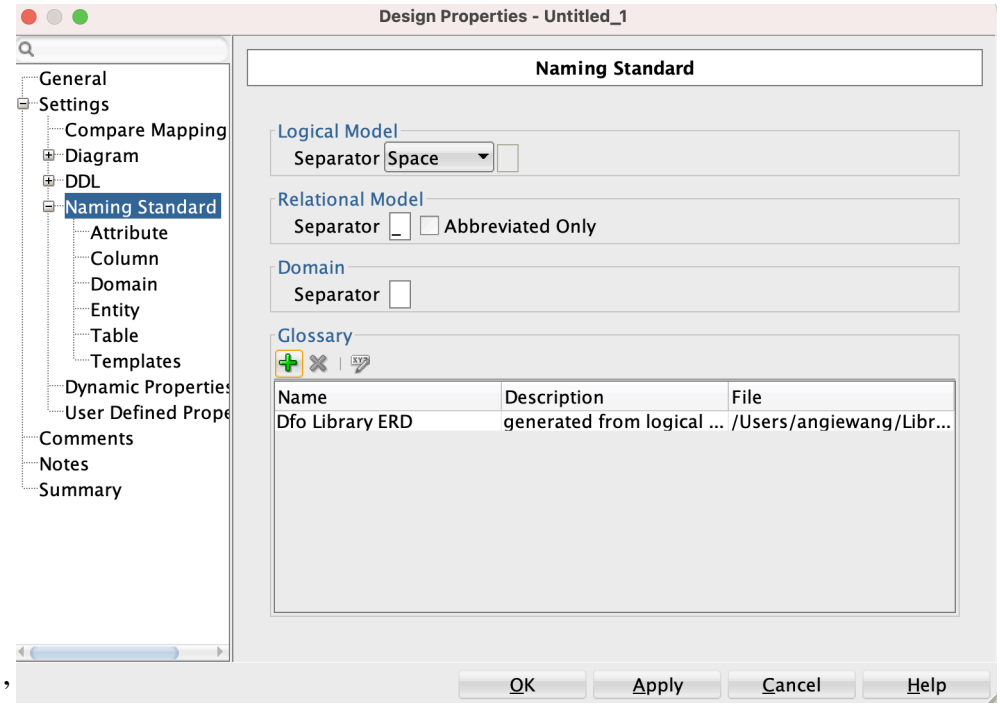
Exercise 2: Forward engineering the design to apply the Glossary and Naming Standard

Overview: In this practice, you forward engineer your design to apply the Glossary and Naming Standard.

Tasks

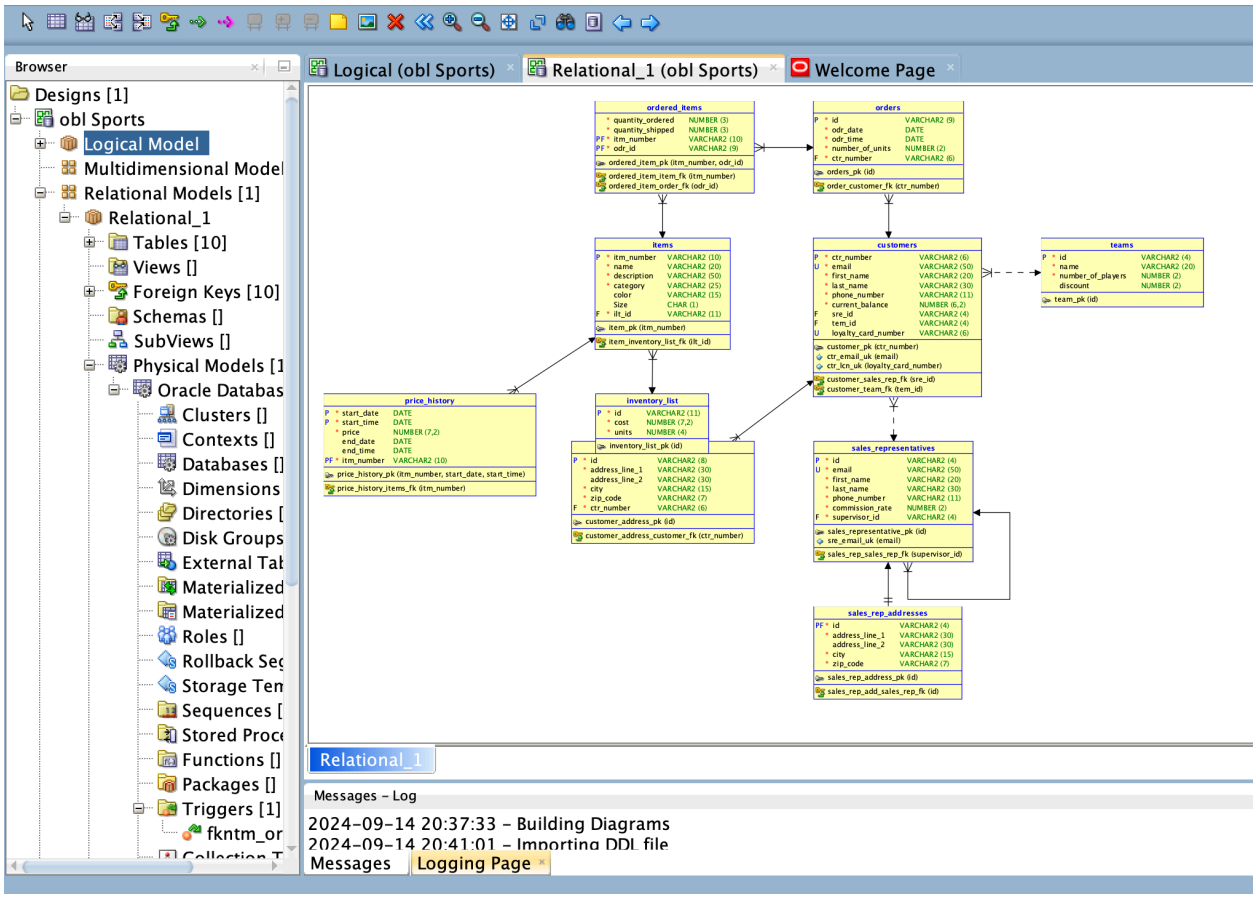
1. For the glossary to be applied during engineering, you must add it on the Naming Standard page in the Preferences dialog box. To ensure that the Glossary gets applied when forward engineering your model perform the following steps:
    a. Right-click the Design model in the Browser and select Properties.
    b. Expand Settings and click the Naming Standard node.
    c. Click the “+” icon in the Glossary region, and navigate to the location of the glossary.





2. To apply the naming standards that you have set in Task 1, perform the following steps:
    a. Click the Engineer (>>) icon in your toolbar. The Engineering to Relational Model dialog box is displayed.

b. Click the General Options tab, and then select the Apply name translation check box. You will notice that the Use preferred abbreviations check box is then highlighted and is selected by default.
c. Click Engineer to engineer your model.



5-2: Mapping Primary and Foreign Keys

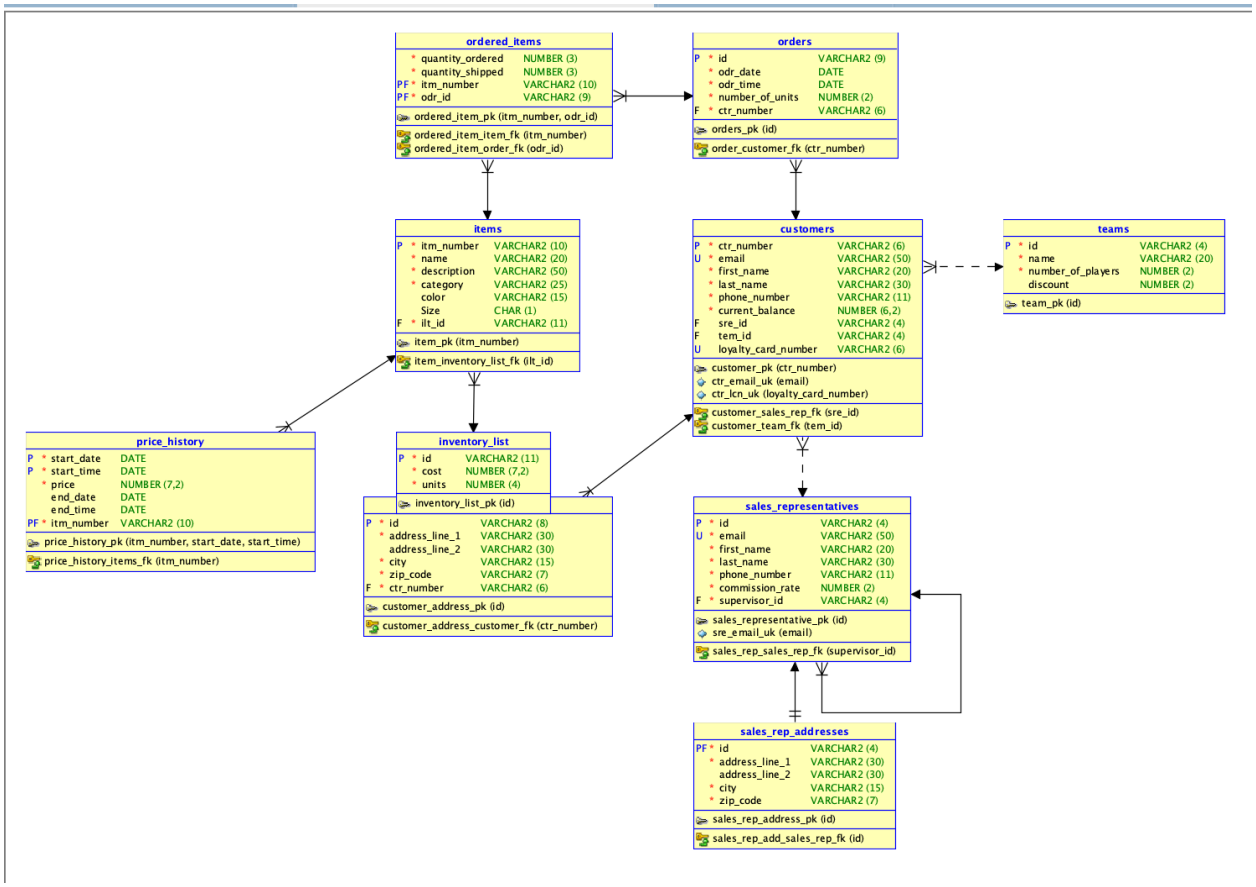
Exercise 1: Observe the mapping of the unique identifiers and relationship in the Relational Model

Overview: In this practice you will observe the mapping of the unique identifiers and relationship in the Academic Database Relational Model.

Tasks

1. Compare the Logical Model and the engineered Relational Model to verify:
    a. The Unique Identifiers that have been mapped as Primary Keys
    b. The Unique Identifiers that have been mapped as Unique Keys

c. The Relationships that have been mapped as Foreign Keys



Exercise 2: Define table name abbreviations in csv file

Overview: In this practice, you define abbreviations for keys and constraints in a .csv file.

Tasks

1. To define the abbreviations for table names, perform the following steps:
    a. Open a spreadsheet application
    b. In the first column list plural table names, and in the second column the required abbreviation for each table.
    c. Save the file as .csv and note the location.

<u>6-1 : Introduction to Oracle Application Express</u>

Exercise 1: Introduction to Oracle Application Express

Overview: In this practice, you will view a document that walks you through the different features of Oracle Application Express.

Tasks

1. Go to Section 0 – Course Resources of the Learner – Learning Path for the course and access the iAcademy APEX Learner Guide.

2. Follow the Guide to learn about the features of Oracle Application Express.

6-2 : Structured Query Language

Exercise 1: Using Help in Oracle Application Express

Overview: In this practice you will:

- Login to Oracle Application Express
- Become familiar with the Help sections in Oracle Application Express

Assumptions: You have been assigned an Oracle Application Express Workspace and the credentials to log in.

Tasks

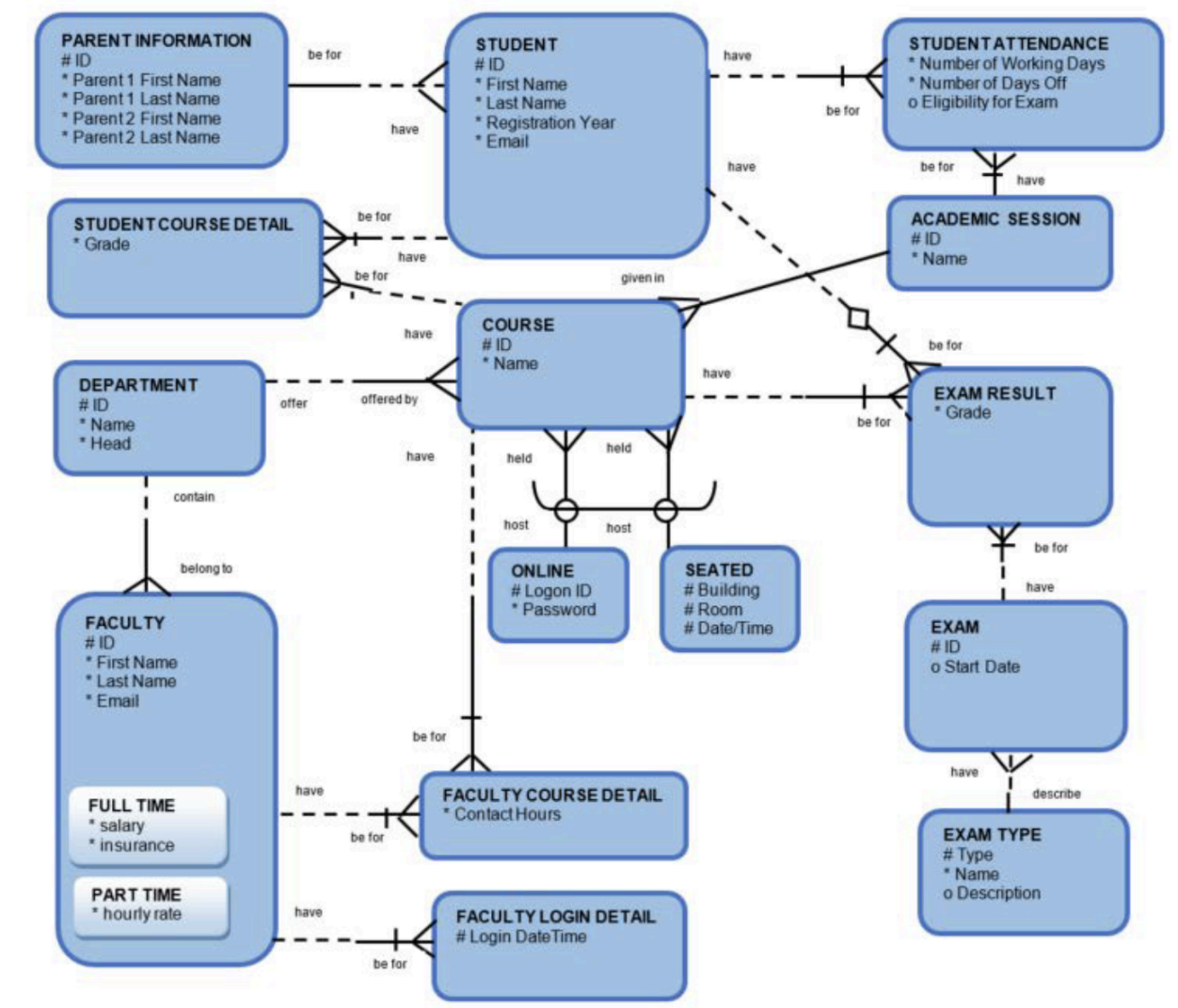
1. Access and log in to Oracle Application Express
2. Click the Help icon, and become familiar with the following section and topics:
    a. Application Express SQL Workshop
        i. *Consist of object browser, SQL commands where you can write SQL codes, store SQL scripts, utilities which consist of query builders, sample datasets, quick SQL and other types of applications*
    b. Managing Database Objects with Object Browser
        i. *Can view tables, indexes, sequences, types, packages, procedures, functions, triggers, database links, materialized views, synonyms, and SODA collections*
    c. Using SQL Commands
        i. *Can write SQL commands*
    d. Using SQL Scripts
        i. *Can store SQL scripts*

6-3 : Defining Data Definition Language (DDL)

Exercise 1: Creating Tables Using Oracle Application Express

Overview: In this practice, you will create the tables for the Academic Database.

Assumptions: The following is the Entity Relationship Diagram (ERD) for the Academic Database where the tables will be created:



1. 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)
2. Run/execute these commands in Oracle Application Express

COURSE

Column Name	Data Type	Nullable	Primary Key
course_id	INT	yes	1
course_name	VARCHAR(100)	no	-
course_grade	VARCHAR (4)	no	-
semester	VARCHAR(20)	no	-
course location	VARCHAR(2)	no	-

FACULTY

Column Name	Data Type	Nullable	Primary Key
professor_id	INT	yes	1
first_name	VARCHAR(50)	no	-
last_name	VARCHAR(50)	no	-
email	VARCHAR(100)	no	-
department	VARCHAR(100)		-

STUDENT

Column Name	Data Type	Nullable	Primary Key
student_id	INT	yes	1
first_name	VARCHAR(50)	no	-
last_name	VARCHAR(50)	no	-
email	VARCHAR(100)	no	-
registration_year	VARCHAR(100)		-

Exercise 2: Altering the Tables

Overview: In this practice, you will

- Alter the tables to set the constraints
- Specify a default value for a column
- Set a table to a read-only status

Assumptions: The primary and foreign key constraints are based on the ERD shown in the previous exercise and the unique constraints are based on the following

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

Tasks

1. Alter the tables in the Academic Database to define the primary key, foreign key and unique constraints.

AD_COURSES

Column Name	Data Type	Constraints
course_id	INT	primary key, auto_increment
course_name	VARCHAR(100)	unique
course_grade	VARCHAR (4)	unique
semester	VARCHAR(20)	
course location	VARCHAR(2)	

AD_STUDENTS

Column Name	Data Type	Constraints
student_id	INT	primary key, auto_increment
first_name	VARCHAR(50)	unique
last_name	VARCHAR(50)	unique
email	VARCHAR(100)	unique
registration_year	VARCHAR(100)	

AD_FACULTY

Column Name	Data Type	Constraints
professor_id	INT	primary key, auto_increment
first_name	VARCHAR(50)	unique
last_name	VARCHAR(50)	unique
email	VARCHAR(100)	unique
department	VARCHAR(100)	foreign key

2. Alter the table AD_FACULTY_LOGIN_DETAILS and specify a default value for the column LOGIN_DATE_TIME of SYSDATE.
    a. ALTER TABLE AD_FACULTY_LOGIN_DETAILS
    MODIFY LOGIN_DATE_TIME TIMESTAMP DEFAULT SYSDATE;
3. Set the AD_PARENT_INFORMATION table to a read-only status.

a. ALTER TABLE AD_PARENT_INFORMATION
READ ONLY;

NOTE: You can execute the INSERT / ALTER TABLE statements in Oracle Application Express in one of the two ways:

Method 1:

a. Open Oracle Application Express and paste the commands into the SQL Commands screen one at a time and run.

Method 2:

a. Open Oracle Application Express and use the same script upload method as you did with the DDL commands above.

Exercise 3: Creating Composite Primary, Foreign and Unique Keys

Overview: In this practice, you will create

- Composite Primary Key
- Composite Foreign Key
- Composite Unique Key
- ** Note – these tables are not a part of the Academic Database

1. Create the `DEPT` table with the following structure:

Column	Data Type	Description
dept_id	number(8)	Department ID
dept_name	varchar2(30)	Department Name
loc_id	number(4)	Location ID

The primary key for this table needs to be defined as a composite comprising of the `dept_id` and `loc_id`.

2. Create the SUPPLIERS and PRODUCTS table with the following structure:

DEPT

Column	Data Type	Constraints
dept_id	number(8)	primary key
dept_name	varchar2(38)	unique
ioc_id	number(4)	foreign key

SUPPLIERS

Column	Data Type	Constraints
sup_id	number(15)	primary key
sup_name	varchar2(30)	unique
contact_name	number(4)	unique

PRODUCTS

Column	Data Type	Constraints
product_id	number(10)	primary key
sup_id	number(15)	foreign key
sup_name	varchar2(30)	unique

SUPPLIERS TABLE

Column	Data Type	Description
sup_id	number(15)	Supplier ID part of composite primary key
sup_name	varchar2(30)	Supplier Name part of composite primary key
contact_name	number(4)	Agent Contact Name

The primary key for this table needs to be defined as a composite comprising of the `sup_id` and `sup_name`.

PRODUCTS TABLE

Column	Data Type	Description
product_id	number(10)	Product ID is the primary key
sup_id	number(15)	Supplier ID that does not hold NULL value
sup_name	varchar2(30)	Supplier Name that does not hold NULL value

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`.

3. Create the `DEPT_SAMPLE` table with the following structure:

Column	Data Type	Description
dept_id	number(8)	Department ID
dept_name	varchar2(30)	Department Name
loc_id	number(4)	Location ID

The UNIQUE key for this table needs to be defined as a composite comprising of the `dept_id` and `dept_name`.

6-4 : Defining Data Manipulation

Exercise 1: Inserting Rows in Tables

Overview: You will insert rows into the tables created for the Academic Database.

Assumptions: The tables have been created for the Academic Database (based on Practice 6-3).
Tasks

1. Insert rows into the tables created for the Academic Database based on the following tables:

AD_ACADEMIC_SESSIONS:

ID	NAME
100	SPRING SESSION
200	FALL SESSION
300	SUMMER SESSION

INSERT INTO *AD_ACADEMIC_SESSIONS (ID, Name)*
VALUES *(400, 'winter session');*

AD_DEPARTMENTS:

ID	NAME	HEAD
10	ACCOUNTING	MARK SMITH
20	BIOLOGY	DAVE GOLD
30	COMPUTER SCIENCE	LINDA BROWN
40	LITERATURE	ANITA TAYLOR

INSERT INTO *AD_DEPARTMENTS (ID, Name, Head)*
VALUES *(50, 'MATH', 'JOHN SMITH');*

AD_PARENT_INFORMATION: (Hint: must return to READ/WRITE status)

ID	PARENT1_FN	PARENT1_LN	PARENT2_FN	PARENT2_LN
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_PARENT_INFORMATION (ID, PARENT1_FN, PARENT1_LN, PARENT2_FN, PARENT2_LN)*
VALUES *(650, 'ROBERT', 'BLUE', 'DAISY', 'BLUE');*

AD_STUDENTS:

ID	FIRST_NAME	LAST_NAME	REG_YEAR	EMAIL	PARENT_ID
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_STUDENTS (ID, FIRST_NAME, LAST_NAME, REG_YEAR, EMAIL, PARENT_ID)*
VALUES *(780, 'ROBERT', 'GARCIA', '01-Mar-2013', 'LGARCIA@SCHOOL.EDU', 620);*

AD_COURSES:

ID	NAME	SESSION_ID	DEPT_ID	LOGON_ID	PASSWORD	BUILDING	ROOM	DATE_TIME
195	CELL BIOLOGY	200	20	-	-	BUILDING D	401	MWF 9-10
190	PRINCIPLES OF ACCOUNTING	100	10	-	-	BUILDING A	101	MWF 12-1
191	INTRODUCTION TO BUSINESS LAW	100	10	-	-	BUILDING B	201	THUR 2-4
192	COST ACCOUNTING	100	10	-	-	BUILDING C	301	TUES 5-7
193	STRATEGIC TAX PLANNING FOR BUSINESS	100	10	TAX123	PASSWORD	-	-	-
194	GENERAL BIOLOGY	200	20	BIO123	PASSWORD	-	-	-

AD_FACULTY:

ID	FIRST_NAME	LAST_NAME	EMAIL	SALARY	INSURANCE	HOURLY_RATE	DEPT_ID
800	JILL	MILLER	JMILL@SCHOOL.EDU	10000	HEALTH	-	20
810	JAMES	BORG	JBORG@SCHOOL.EDU	30000	HEALTH,DENTAL	-	10
820	LYNN	BROWN	LBROWN@SCHOOL.EDU	-	-	50	30
830	ARTHUR	SMITH	ASMITH@SCHOOL.EDU	-	-	40	10
840	SALLY	JONES	SJONES@SCHOOL.EDU	50000	HEALTH,DENTAL,VISION	-	40

AD_EXAM_TYPES:

TYPE	NAME	DESCRIPTION
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_EXAM_TYPES (TYPE, NAME, DESCRIPTION)*
VALUES *('PRJ', 'Project', 'DEMONSTRATE UNDERSTANDING THROUGH A PROJECT');*

AD_EXAMS:

ID	START_DATE	EXAM_TYPE	COURSE_ID
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 A*D_EXAMS (ID, START_DATE, EXAM_TYPE, COURSE_ID)*
VALUES *(550, '10-Oct-2014', 'MCE', 195);*

AD_EXAM_RESULTS:

STUDENT_ID	COURSE_ID	EXAM_ID	EXAM_GRADE
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_EXAM_RESULTS (STUDENT_ID, COURSE_ID, EXAM_ID, EXAM_GRADE)*
VALUES *(770, 195, 550, 88);*

AD_STUDENT_ATTENDANCE:

STUDENT_ID	SESSION_ID	NUM_WORK_DAYS	NUM_DAYS_OFF	EXAM_ELIGIBILITY
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_ATTENDANCE (STUDENT_ID, SESSION_ID, NUM_WORK_DAYS, NUM_DAYS_OFF, EXAM_ELIGIBILITY)*
VALUES *(780, 300, 180, 10, 'Y');*

AD_STUDENT_COURSE_DETAILS:

STUDENT_ID	COURSE_ID	GRADE
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_STUDENT_COURSE_DETAILS (STUDENT_ID, COURSE_ID, GRADE)*
VALUES *(780, 195, 'B');*

AD_FACULTY_COURSE_DETAILS:

FACULTY_ID	COURSE_ID	CONTACT_HRS
800	192	3
800	193	4
800	190	5
800	191	3
810	194	4
810	195	5

INSERT INTO *AD_FACULTY_COURSE_DETAILS (FACULTY_ID, COURSE_ID, CONTACT_HRS)*
VALUES *(820, 191, 2);*

AD_FACULTY_LOGIN_DETAILS:

FACULTY_ID	LOGIN_DATE_TIME
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

INSERT INTO AD_FACULTY_LOGIN_DETAILS (FACULTY_ID, LOGIN_DATE_TIME) VALUES (830, TO_TIMESTAMP('01-JUN-17 05.15.00.000000 PM', 'DD-MON-YY HH.MI.SS.FF AM'));

Note: You can write the INSERT statements and save them as .sql script which can then be uploaded into APEX and executed.

You can run/execute these commands in Oracle Application Express as a script:

a. Save the above DDL statements as a text file.
b. Logon to APEX
c. Click SQL Workshop
d. Click SQL Scripts
e. Click the Upload button
f. In the Upload Script window, click Browse..., choose the SQL script; provide a Script Name. Once done, click Upload.
g. You will get a message that the script is uploaded. Click the Run button. You are taken to a window that displays the script uploaded. Click the Run Script button, to execute the statements.
h. The Results window displays whether the statements were executed successfully.

Exercise 2: Updating Rows in the Tables

Overview: You will update the records in FACULTY_LOGIN_DETAILS table to include a DETAILS field in the table.

Tasks

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.

UPDATE AD_FACULTY_LOGIN_DETAILS
SET DETAILS = 'Default detail'
WHERE FACULTY_ID IN (800, 810, 840, 820, 830);

2. Update at least 2 records in the DETAILS column in the faculty login details table.

-- Update the DETAILS column for specific records in FACULTY_LOGIN_DETAILS

UPDATE FACULTY_LOGIN_DETAILS
SET DETAILS = 'Updated detail 1'
WHERE faculty_id = 1; -- Specify the condition to select the first record

UPDATE FACULTY_LOGIN_DETAILS
SET DETAILS = 'Updated detail 2'
WHERE faculty_id = 2; -- Specify the condition to select the second record

**Note: You will have to look up the LOGIN_DATE_TIME values for the records being updated since it is part of the primary key.

Verify that the DETAILS column has been updated with the values:
SELECT faculty_id, DETAILS
FROM FACULTY_LOGIN_DETAILS
WHERE faculty_id IN (1, 2); -- Use the same IDs you updated

<u>6-5: Defining Transaction Control</u>

Exercise 1: Controlling Transactions

Overview: In this practice you will control transactions using the following statements:

- COMMIT
- ROLLBACK
- SAVEPOINT

Assumptions

- Since TCL statements cannot be executed in Oracle Application Express (APEX) you will create the command language but not actually execute it

Tasks

1. Suppose a table with this structure is created:

```
CREATE TABLE AD_STUDENT_TEST_DETAILS
  (
    STUDENT_ID            NUMBER NOT NULL ,
    FIRST_NAME            VARCHAR2(50) ,
    STUDENT_REG_YEAR      DATE
  ) ;
```

a. Then the table is altered to add an email_addr column:

```
ALTER TABLE AD_STUDENT_TEST_DETAILS ADD ( EMAIL_ADDR VARCHAR2(100)
UNIQUE );
```

b. After the ALTER a Savepoint is created called ALTER_DONE.
c. A ROLLBACK is issued after the Savepoint ALTER_DONE. Would the new email field still be there?

The new email field would not still be there. Any ROLLBACK that was issued before the savepoint ALTER_DONE would be saved however anything after it would be deleted. Since the ROLLBACK is issued after the Savepoint ALTER_DONE, then the new email field will be deleted.

The EMAIL_ADDR column will no longer exist after the rollback. The ROLLBACK TO ALTER_DONE will reverse the ALTER TABLE statement.

2. If an INSERT is done to add rows into the test table and a Savepoint is then created called INSERT_DONE.
    a. Then an UPDATE to a row in the test table is done and a Savepoint is created called UPDATE_DONE.
    b. 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?
    c. 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?

```
INSERT INTO AD_STUDENT_TEST_DETAILS VALUES(920, 'MAC', TO_DATE('01-JAN-2012','DD-MON-YYYY'),NULL);
INSERT INTO AD_STUDENT_TEST_DETAILS VALUES(940, 'RUTH', TO_DATE('01-SEP-2012','DD-MON-YYYY'),NULL);
INSERT INTO AD_STUDENT_TEST_DETAILS VALUES(950, 'ROBERT', TO_DATE('01-MAR-2012','DD-MON-YYYY'),NULL);
INSERT INTO AD_STUDENT_TEST_DETAILS VALUES(960, 'JEANNE', TO_DATE('01-MAR-2012','DD-MON-YYYY'),NULL);

SAVEPOINT CREATE_DONE;

UPDATE AD_STUDENT_TEST_DETAILS
SET EMAIL_ADDR = 'Mac@abc.com'
WHERE STUDENT_ID = 940;

SAVEPOINT UPDATE_DONE;

DELETE FROM AD_STUDENT_TEST_DETAILS WHERE STUDENT_ID = 950;

SAVEPOINT DELETE_DONE;

ROLLBACK TO UPDATE_DONE;
```

After rolling back to the `UPDATE_DONE` savepoint, the table will include all rows inserted before the `INSERT_DONE` savepoint. The row deleted after the `DELETE_DONE` savepoint will be restored, while the updates made up to the `UPDATE_DONE` savepoint will be retained.

<u>6-6 : Retrieving Data</u>

Exercise 1: Retrieving Columns from tables

Overview: In this practice you will perform the following tasks on the tables created for the Academic Database:

- Select all the columns from a table
- Select specific columns from a table

Tasks

1. Write a simple query to view the data inserted in the tables created for the academic database
    a. *SELECT * FROM AD_COURSES;*
    b. *SELECT * FROM AD_FACULTY;*
    c. *SELECT * FROM AD_STUDENTS;*
2. Write a query to retrieve the exam grade obtained by each student for every exam attempted.

```
SELECT
    student_id
FROM
    AD_STUDENTS
JOIN
    AD_EXAM_GRADES ON student_id = student_id
JOIN
    AD_EXAMS ON exam_id = exam_id
ORDER BY
    student_id, exam_id
```

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
    student_id,
    attendance_required_days
    COUNT(class_date) AS attended_days,
    CASE
        WHEN COUNT(class_date) >= attendance_required_days THEN 'Eligible'
        ELSE 'Not Eligible'
    END AS eligibility_status
FROM
    AD_STUDENTS
JOIN
    AD_ATTENDANCE ON student_id = student_id
JOIN
    AD_EXAMS ON course_id = course_id
WHERE
    attended = TRUE
GROUP BY
    student_id,
    attendance_required_days
ORDER BY
```

student_id,
exam_id;

4. Display the LOGIN_DATE_TIME for each faculty member.

```
SELECT
  faculty_id,
  email,
  LOGIN_DATE_TIME
FROM
  AD_FACULTY_LOGIN_DETAILS
ORDER BY
  faculty_id;
```

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

```
SELECT
  department_name,
FROM
  AD_DEPARTMENTS
JOIN
  AD_FACULTY ON head_faculty_id = faculty_id
ORDER BY
  department_name;
```

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 formatted_output
FROM
  AD_STUDENTS;
```

7. Display all the distinct exam types from the AD_EXAMS table.

```
SELECT DISTINCT exam_type
FROM AD_EXAMS;
```

<u>6-7 : Restricting Data Using WHERE Statement</u>

Exercise 1: Restricting Data Using SELECT

Overview: In this practice you limit the rows displayed with:

- The WHERE clause
- The comparison operators
- Logical conditions using AND, OR, and NOT operators

Tasks

1. Display the course details for the Spring Session.

```
SELECT
  course_id,
  course_name,
  course_description
FROM
  AD_COURSES
JOIN
  AD_COURSE_SESSIONS ON course_id = course_id
```

```
JOIN
   AD_ACADEMIC_SESSIONS ON session_id = session_id
WHERE
   session_name = 'Spring';
```

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

```
SELECT
   student_id,
   first_name,
   last_name,
   email,
   grade
FROM
   AD_STUDENTS
JOIN
   AD_EXAM_GRADES ON student_id = student_id
WHERE
   grade > 95;
```

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

```
SELECT
   student_id,
   first_name,
   last_name,
   email,
   grade
FROM
   AD_STUDENTS
JOIN
   AD_EXAM_GRADES ON student_id = student_id
WHERE
   grade BETWEEN 65 AND 70;
```

4. Display the students who registered after 01-Jun-2012.

```
SELECT
   student_id,
   registration_date
FROM
   AD_STUDENTS
WHERE
   registration_date > DATE '2012-06-01'
ORDER BY
   registration_date;
```

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

```
SELECT
   course_id,
   course_name,
   department_id,
FROM
   COURSES
WHERE
```

```
    department_id IN (10, 30);
```

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

```
SELECT
    student_id,
    first_name,
FROM
    AD_STUDENTS
WHERE
    first_name LIKE 'J';
```

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

```
SELECT
    student_id,
FROM
    STUDENTS
JOIN
    ENROLLMENTS ON student_id = student_id
WHERE
    course_id IN (190, 193);
```

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

```
SELECT
    course_id,
    department_id,
    session_id,
    course_description
FROM
    COURSES
JOIN
    SESSIONS ON course_id = course_id
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_id,
    session_id,
    course_description
FROM
    COURSES
LEFT JOIN
    SESSIONS ON course_id = course_id
WHERE
    session_id IS NULL
    OR session_id NOT IN (200, 300);
```

10. Display the course details for department 20

```
SELECT
    course_id,
    department_id,
```

```
    course_description
FROM
    COURSES
WHERE
    department_id = 20;
```

6-8 : Sorting Data Using ORDER BY

Exercise 1: Sorting Data Using ORDER BY

Overview: In this practice you will:

- Sort rows by using the ORDER BY clause

Tasks

1. 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;
    ```
    b. AD_EXAM_RESULTS ordered by STUDENT_ID and COURSE_ID
    ```
    SELECT *
    FROM AD_EXAM_RESULTS
    ORDER BY STUDENT_ID, COURSE_ID;
    ```
    c. AD_STUDENT_ATTENDANCE ordered by STUDENT_ID
    ```
    SELECT *
    FROM AD_STUDENT_ATTENDANCE
    ORDER BY STUDENT_ID;
    ```
    d. AD_DEPARTMENTS ordered by the department ID
    ```
    SELECT *
    FROM AD_DEPARTMENTS
    ORDER BY DEPARTMENT_ID;
    ```
2. Display the percentage of days students have taken days off and sort the records based on the percentage calculated.
    ```
    SELECT
        student_id,
        total_days,
        days_off,
        (days_off / total_days * 100) AS percentage_days_off
    FROM
        AD_STUDENT_ATTENDANCE
    WHERE
        total_days > 0
    ORDER BY
        percentage_days_off DESC;
    ```
3. Display the top 5 students based on exam grade results.
    ```
    SELECT
        student_id,
        first_name,
        last_name,
    ```

```
    MAX(exam_grade) AS highest_exam_grade
FROM
    AD_STUDENTS
JOIN
    AD_EXAM_RESULTS ON student_id = student_id
GROUP BY
    student_id, first_name, last_name
ORDER BY
    highest_exam_grade DESC
FETCH FIRST 5 ROWS ONLY;
```

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

```
SELECT
    parent_id,
    parent_name,
    contact_number,
    email
FROM
    AD_PARENTS
ORDER BY
    parent_id;
```

6-9 : Joining Tables Using JOIN

Exercise 1: Using JOINS in SQL Queries

Overview: In this practice you:

- Access data from more than one table using equijoins and non-equijoins
- Use OUTER joins to view data that generally does not meet a join condition
- Generate a Cartesian Product

Tasks

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

```
SELECT
    department_id,
    course_id,
FROM
    COURSES
JOIN
    DEPARTMENTS ON department_id = department_id
ORDER BY
    department_id, course_id;
```

2. Display the courses offered in the Fall session.

```
SELECT
   course_id,
   course_name,
FROM
    COURSES
JOIN
    SESSIONS ON course_id = course_id
WHERE
    session_id = #; –whatever the number is to represent the fall session
```

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

```
SELECT
    course_id,
    department_id,
    student_id,
FROM
    COURSES
JOIN
    DEPARTMENTS ON department_id = department_id
JOIN
    ENROLLMENTS ON course_id = course_id
JOIN
    STUDENTS ON student_id = student_id
ORDER BY
    course_id, student_id;
```

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

```
SELECT
    course_id,
    department_id,
    student_id,
FROM
    COURSES
JOIN
    DEPARTMENTS ON department_id = department_id
JOIN
    ENROLLMENTS ON course_id = course_id
JOIN
    STUDENTS ON student_id = student_id
WHERE
    department_id = 20
ORDER BY
    course_id, student_id;
```

5. 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
    student_id,
    course_id,
    exam_grade
FROM
    EXAM_RESULTS
WHERE
    course_id BETWEEN 190 AND 192;
```

6. Retrieve the rows from the AD_EXAM_RESULTS table even if there are no matching records in the AD_COURSES table.

```
SELECT
    student_id,
```

course_id,
exam_grade,
course_name,
department_id
FROM
AD_EXAM_RESULTS
LEFT JOIN
AD_COURSES ON course_id = course_id;

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

```
SELECT *  FROM AD_EXAMS
CROSS JOIN AD_EXAM_TYPES;
```

All possible combinations of rows from the AD_EXAMS table with rows from the AD_EXAM_TYPES table.