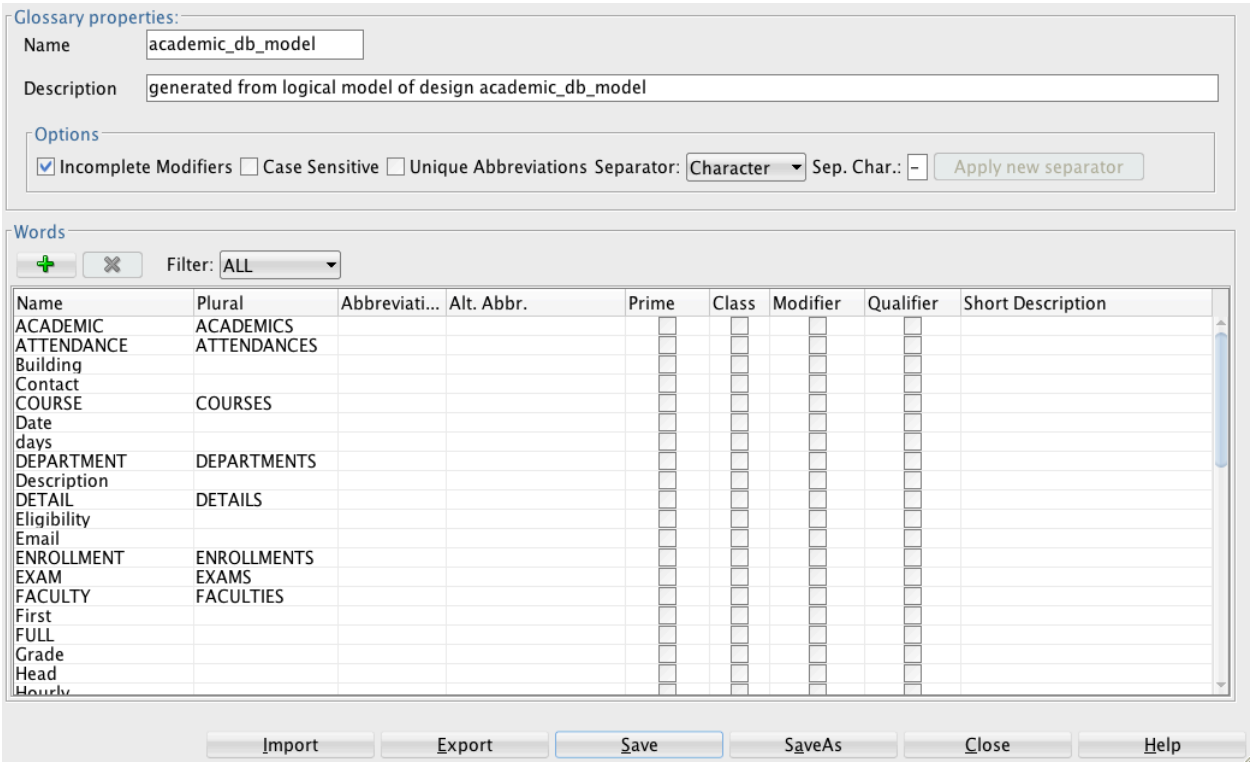
**Name**: Danny Chen
**Assignment**: SQL Database Foundations Sections 5, 6: Mapping to Physical Model, SQL Syntax Intro
**Date:** 09/15/24

**NOTE:** Some tasks involve just following procedures. For their answers, screenshot(s) are posted that shows the end result of following the procedure.

**5.1**
**Exercise 1**: Create a Glossary from the Academic Database Logical Model

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

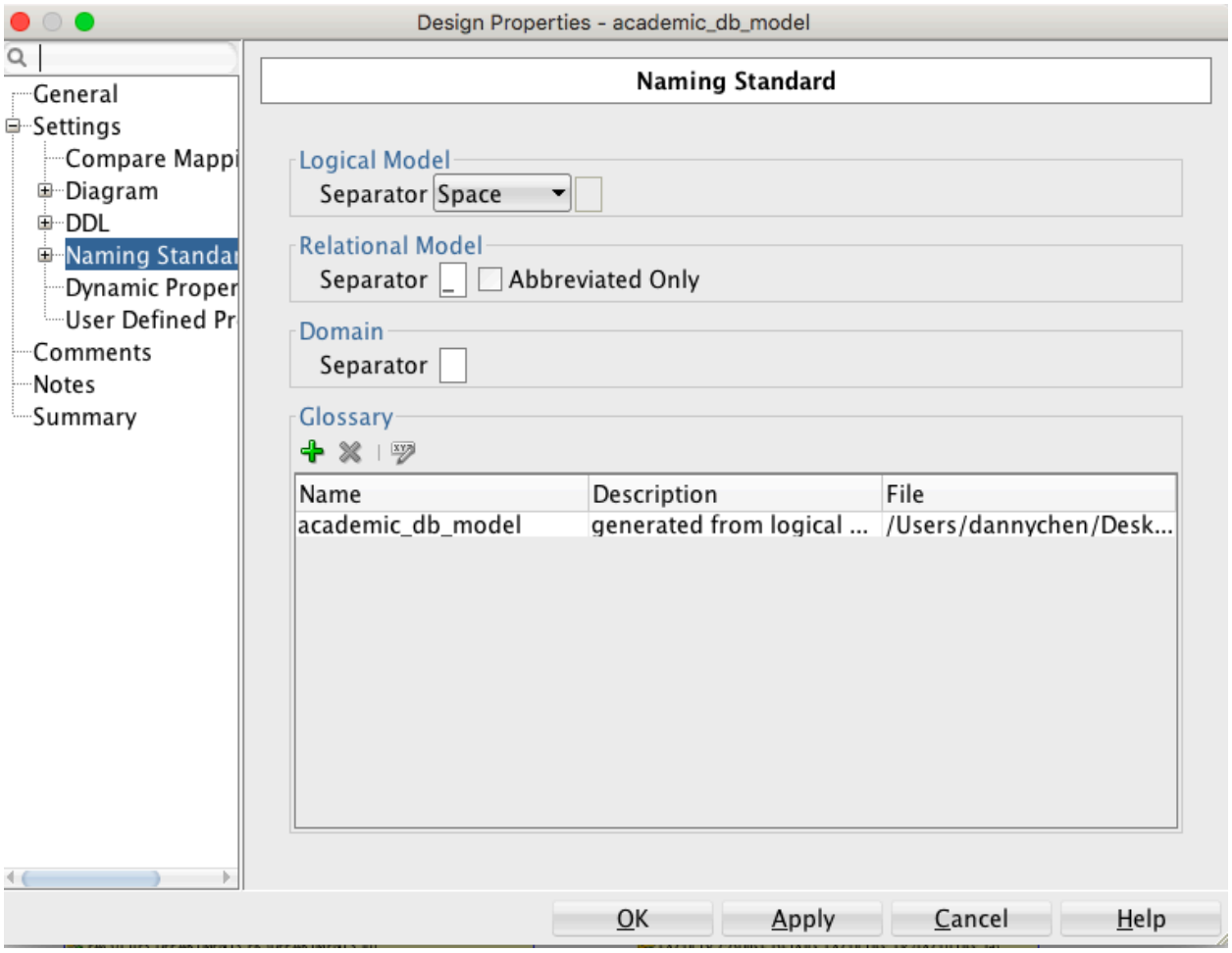
**ANSWER:**




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

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

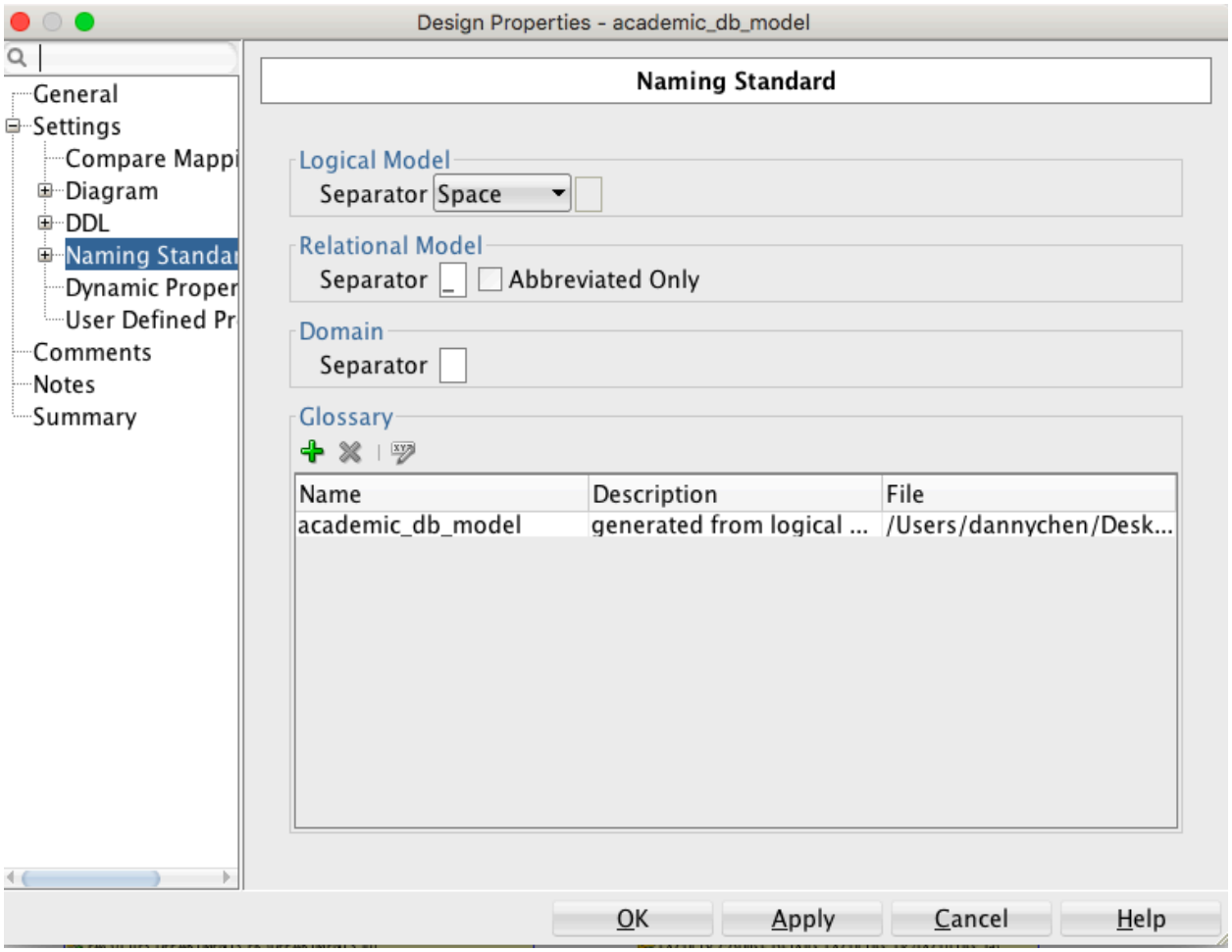
**ANSWER**:




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

**ANSWER**:

1. Naming standards have been applied per screenshot below




2. Click here to see engineered model

**5.2**

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

1. Compare the Logical Model and the engineered Relational Model to verify:
    a. The Unique Identifiers that have been mapped as Primary Keys
        i. All tables that have "id" as a column have become a primary key as indicated by "P" in the engineered relational model.
        ii. Some tables have keys that act as both primary and foreign keys, mainly involving the M:M relationships that are resolved with an intersection entity and multiple 1:M barred relationships. Specifically, these tables are ENROLLMENTS, STUDENT_ATTENDANCES, EXAM_RESULTS, FACULTY_LOGIN_DETAILS, and FACULTY_COURSE_DETAILS.
    b. The Unique Identifiers that have been mapped as Unique Keys

i. Both the STUDENTS' and FACULTIES' email columns are unique keys.

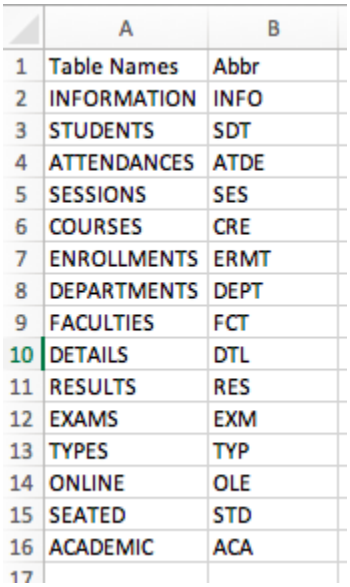
c. The Relationships that have been mapped as Foreign Keys
   i. The table that is on the M (Many) side of the relationship usually has the foreign key. For example, PARENT_INFORMATION and STUDENTS have a 1:M relationship, so STUDENTS has a foreign key referencing the PARENT_INFORMATION's id.
   ii. As noted before, some tables have keys that act as both primary and foreign keys.

**Exercise 2**: Define table name abbreviations in csv file

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

**ANSWER** (screenshot below):

- NOTES:
  - The instructions mention to list plural table names for the first column, but some table names do not have a plural form but were included anyway
  - The abbreviated form for table names that were more than one word did not appear to work. For example, PRIN and ACSE would not replace PARENT_INFORMATION and ACADEMIC_SESSION respectively. However, it did work when I gave an abbreviation for each word of a table name like INFO for INFORMATION and ACA for ACADEMIC.



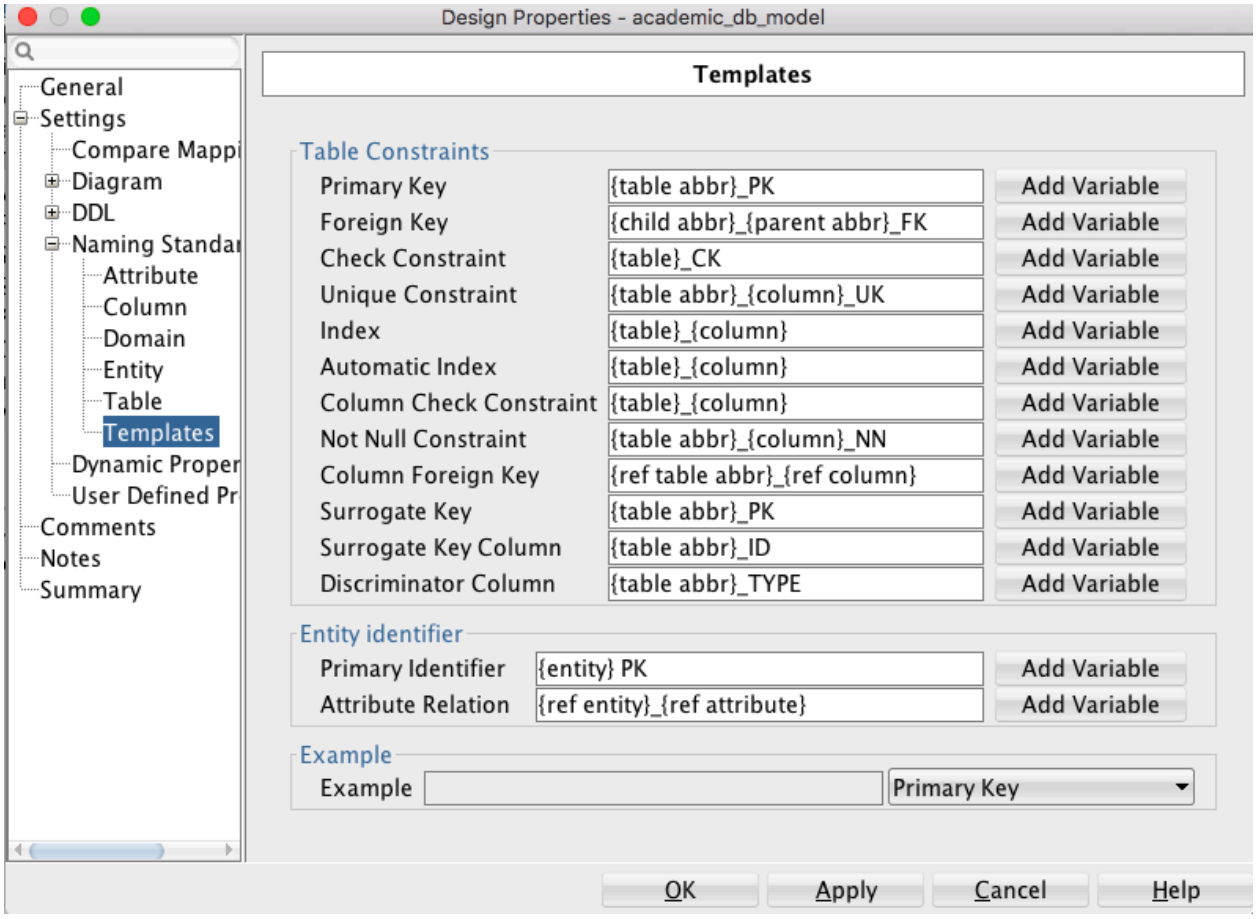
| | A | B |
|---|---|---|
| 1 | Table Names | Abbr |
| 2 | INFORMATION | INFO |
| 3 | STUDENTS | SDT |
| 4 | ATTENDANCES | ATDE |
| 5 | SESSIONS | SES |
| 6 | COURSES | CRE |
| 7 | ENROLLMENTS | ERMT |
| 8 | DEPARTMENTS | DEPT |
| 9 | FACULTIES | FCT |
| 10 | DETAILS | DTL |
| 11 | RESULTS | RES |
| 12 | EXAMS | EXM |
| 13 | TYPES | TYP |
| 14 | ONLINE | OLE |
| 15 | SEATED | STD |
| 16 | ACADEMIC | ACA |
| 17 | | |

**Exercise 3**: Define name template

1. You can set a template for the keys, indexes, and constraints in table or entity by using combinations of predefined variables. To define the name patterns perform the following steps:
    a. Right-click the Academic Database design in the Object Browser and select **Properties**. Expand **Settings > Naming Standard** and select **Templates**.
    b. Set the predefined variables as follows: **Table Constraints**

| Element | Pre-Defined Variable |
|---|---|
| Primary Key | `{table abbr}_PK` |
| Foreign Key | `{child abbr}_{parent abbr}_FK` |
| Check Constraint | `{table}_CK` |
| Unique Constraint | `{table abbr}_{column}_UK` |
| Index | `{table}_{column}` |
| Automatic Index | `{table}_{column}` |
| Column Check Constraint | `{table}_{column}` |
| Not Null Constraint | `{table abbr}_{column}_NN` |
| Column Foreign Key | `{ref table abbr}_{ref column}` |
| Surrogate Key | `{table abbr}_PK` |
| Surrogate Key Column | `{table abbr}_ID` |
| Discriminator Column | `{table abbr}_TYPE` |

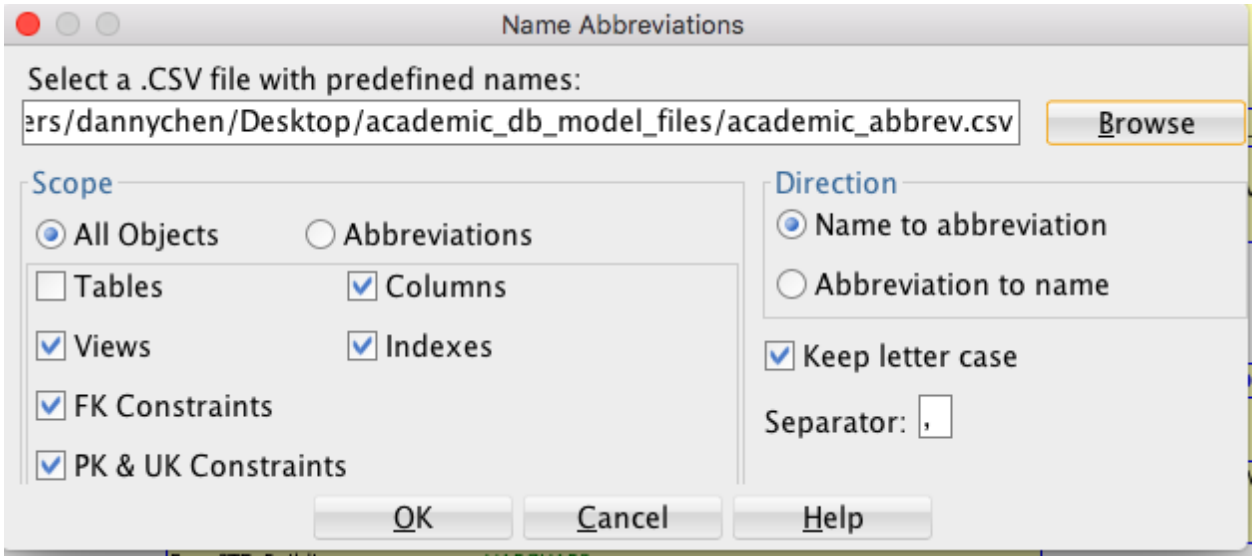
**ANSWER**:




**Exercise 4**: Apply Name Template to the Relational Model

1. To apply the template to the entire Relational model, perform the following:
    a. Click Tools > Name abbreviations.
    b. Browse to the .csv file containing the abbreviations.
    c. Un-check Tables (to maintain existing names from the Glossary), and then click OK.

**ANSWER**:

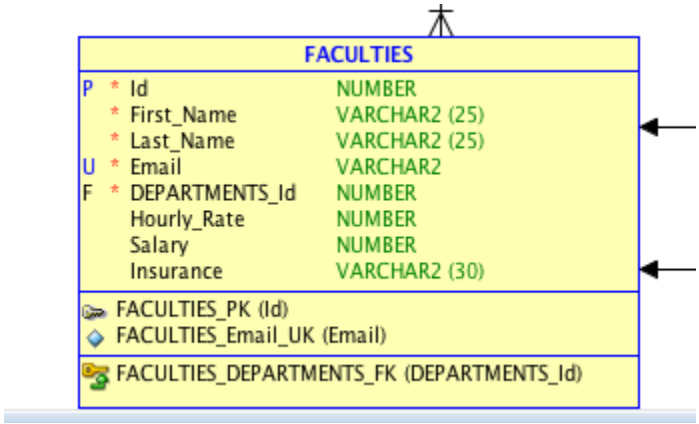



**Exercise 5:** Select how subtypes are generated in the Relational Model

1. To define how subtypes are mapped to the Academic Database Relational Model, perform the following steps:
    a. Click the Logical tab.
    b. Double click the Faculty Super type entity to edit properties
    c. Select Subtypes from the Options in the Left pane.
    d. From the Subtree Generation drop down option, select Single Table. Click OK.
    e. Re-engineer to Relational Model.

**ANSWER**:

- The relational model looks the same as the one from 5_1 practice. The only difference is that the subtypes' attributes are now combined with supertype FACULTIES' attribute




**6.1**

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

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.

**ANSWER**:

- What I learned about Oracle Application Express
    - Oracle Application Express is a web application, deployment, and maintenance tool
    - Requires minimal programming knowledge
    - Allows creating a database web application that's reliable, scalable, and secure
    - It has an SQL workshop, application builder, and object browser

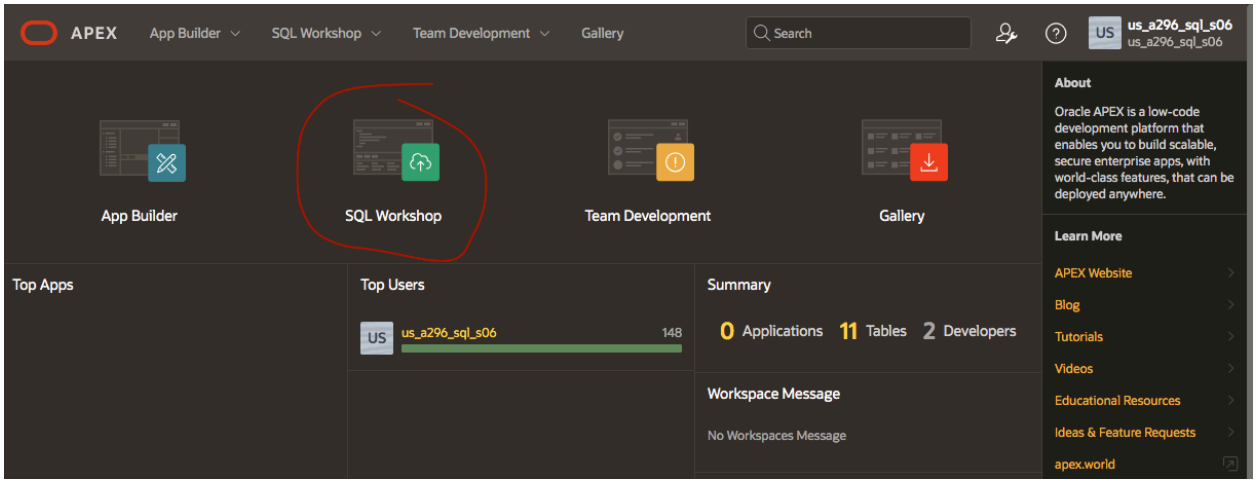
**6.2**

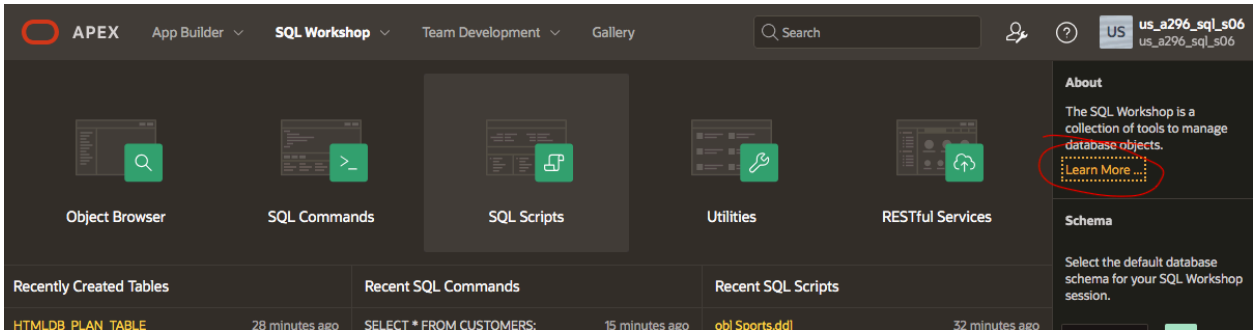
**Exercise 1:** Using Help in Oracle Application Express

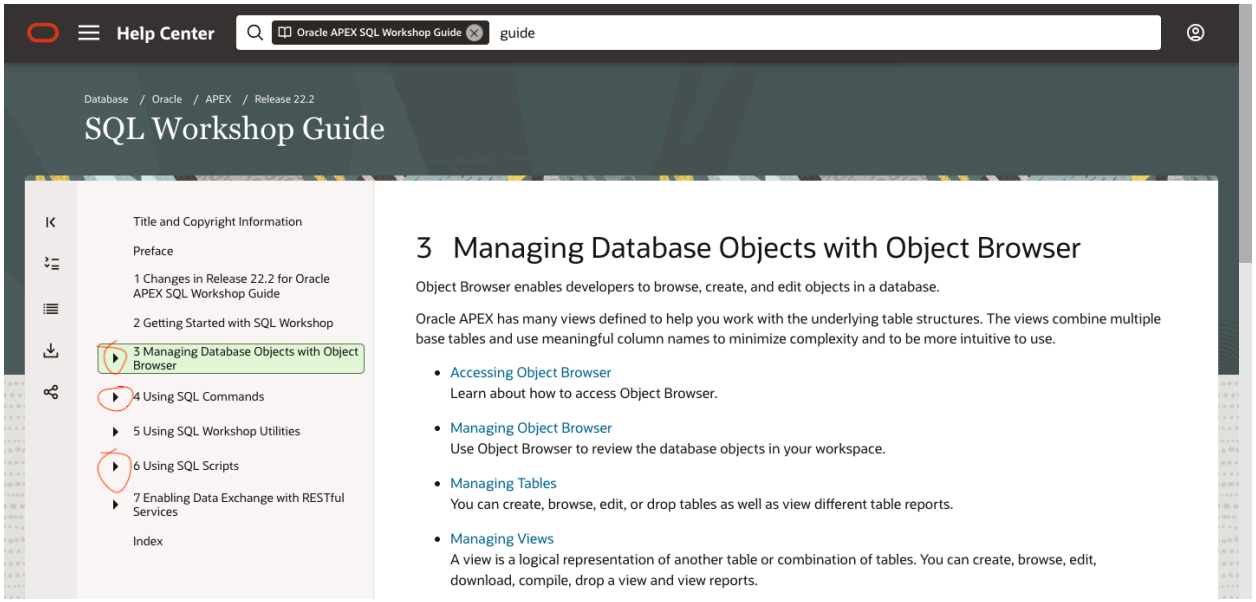
1. Access and log in to Oracle Application Express
2. Click the Help icon, and become familiar with the following section and topics:
    a. Application Express SQL Workshop
        i. Managing Database Objects with Object Browser
        ii. Using SQL Commands
        iii. Using SQL Scripts

**ANSWER**:

1. Clicked on the "SQL workshop" tab
2. Click "Learn More" on the About section at the top right of the page
3. This will open a page to the Help Center and show the "SQL Workshop Guide" that has the exact topics mentioned in the Exercise



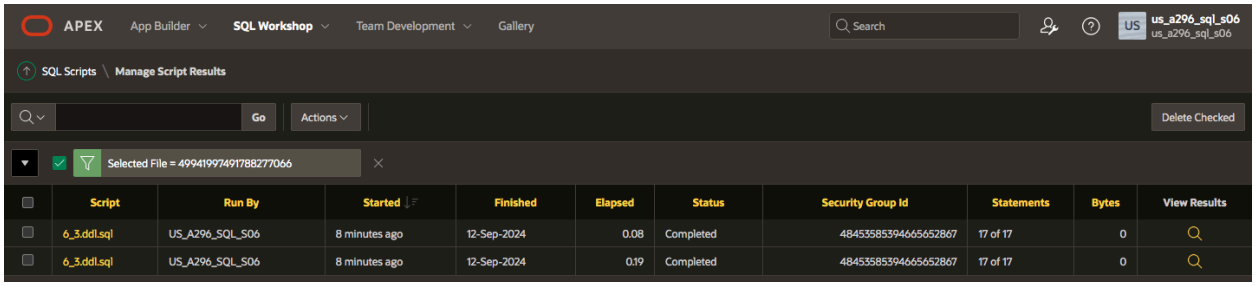



**6.3**

**Exercise 1:** Creating Tables Using Oracle Application Express

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)
    a. **ANSWER**: Click here to see the DDL statements (a ddl file) used to create the table
2. Run/execute these commands in Oracle Application Express
    a. **ANSWER**: Below is the image of the script that was run. It was run twice as I fixed the errors that came from the first run.




**Exercise 2**: Altering the Tables

1. Alter the tables in the Academic Database to define the primary key, foreign key and unique constraints.
2. Alter the table AD_FACULTY_LOGIN_DETAILS and specify a default value for the column LOGIN_DATE_TIME of SYSDATE.
3. Set the AD_PARENT_INFORMATION table to a read-only status.

**ANSWER**:

- Click here to see the statements used to add constraints, default values, and read-only statuses.

Exercise 3: Creating Composite Primary, Foreign and Unique Keys

1. Create the DEPT table with the following structure. The primary key for this table needs to be defined as a composite consisting of the dept_id and loc_id.

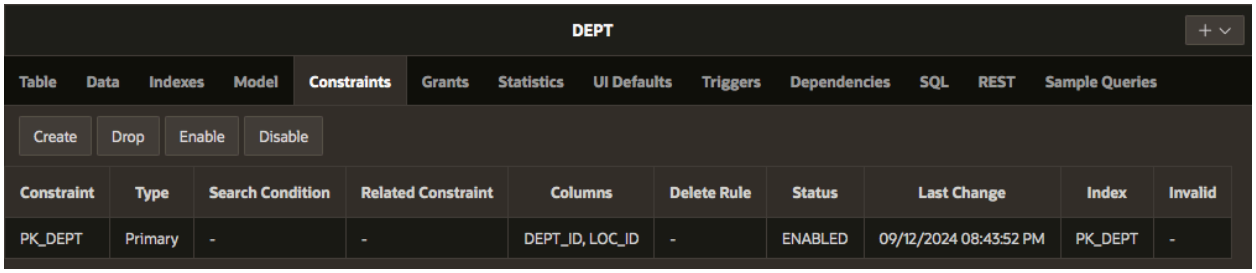
| Column | Data Type | Description |
|---|---|---|
| dept_id | number(8) | Department ID |
| dept_name | varchar2(30) | Department Name |
| loc_id | number(4) | Location ID |

**ANSWER**:

```
CREATE TABLE DEPT (
   dept_id NUMBER(8),
   dept_name VARCHAR2(30),
   loc_id NUMBER(4),
   CONSTRAINT pk_dept PRIMARY KEY (dept_id, loc_id)
);
```

Add Column | Modify Column | Rename Column | Drop Column | Rename | Copy | Drop | Truncate | Create Lookup Table | Create App

| Column Name | Data Type | Nullable | Default | Primary Key |
|---|---|---|---|---|
| DEPT_ID | NUMBER(8,0) | No | - | 1 |
| LOC_ID | NUMBER(4,0) | No | - | 2 |
| DEPT_NAME | VARCHAR2(30) | Yes | - | - |



DEPT

Table | Data | Indexes | Model | Constraints | Grants | Statistics | UI Defaults | Triggers | Dependencies | SQL | REST | Sample Queries

Create | Drop | Enable | Disable

| Constraint | Type | Search Condition | Related Constraint | Columns | Delete Rule | Status | Last Change | Index | Invalid |
|---|---|---|---|---|---|---|---|---|---|
| PK_DEPT | Primary | - | - | DEPT_ID, LOC_ID | - | ENABLED | 09/12/2024 08:43:52 PM | PK_DEPT | - |

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

**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**

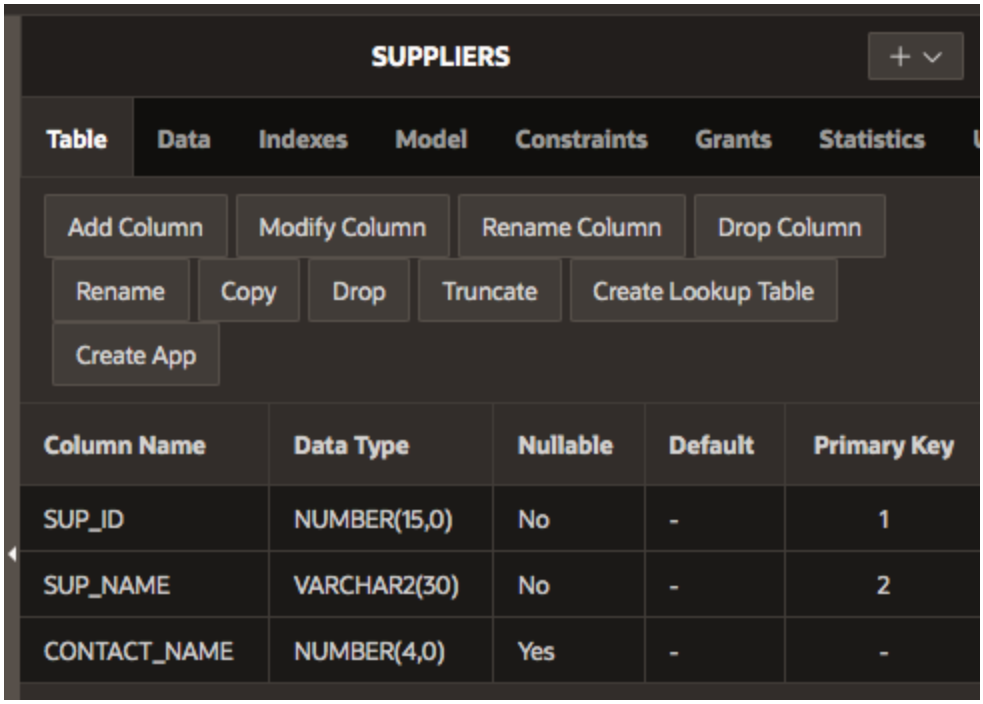
| Column | Data Type | Description |
|---|---|---|
| product_id | number(10) | Product ID is the primary key |
| sup_id | number(15) | Supplier ID that does not hold NULL value |
| sup_name | varchar2(30) | Supplier Name that does not hold NULL value |

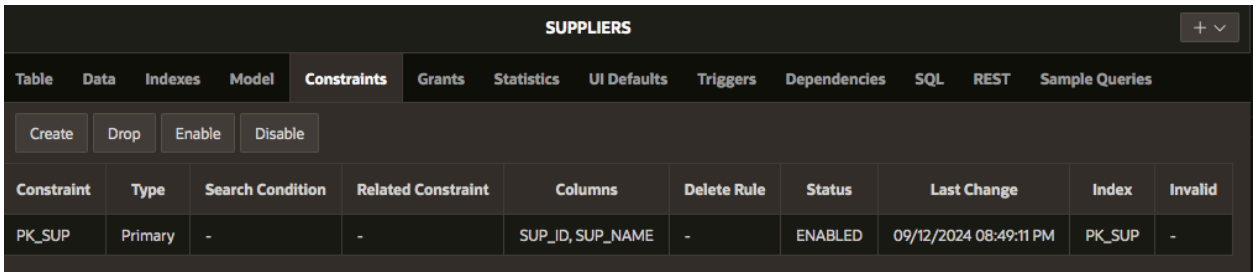
The primary key for this table is `product_id`. The foreign key for this table needs to be defined as a composite comprising of the `sup_id` and `sup_name`.

**ANSWER**:

SUPPLIERS table

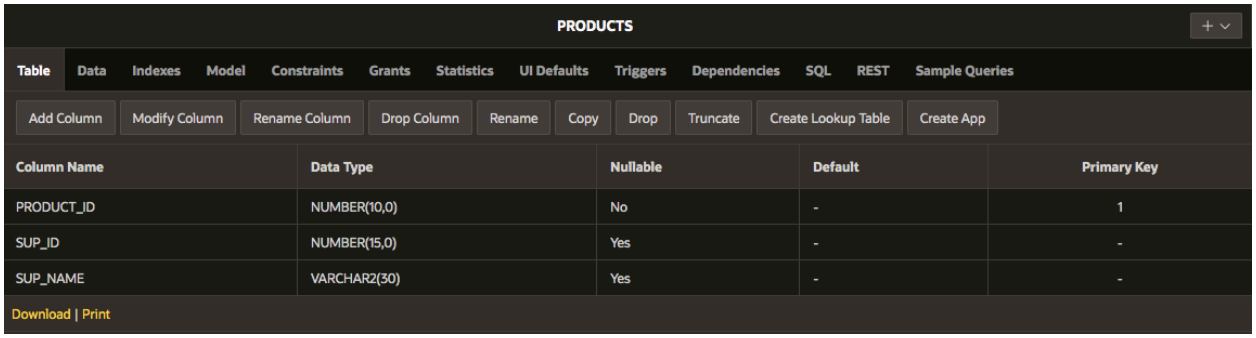
```
CREATE TABLE SUPPLIERS (
   sup_id NUMBER(15),
   sup_name VARCHAR2(30),
   contact_name NUMBER(4),
   CONSTRAINT pk_sup PRIMARY KEY (sup_id, sup_name)
);
```

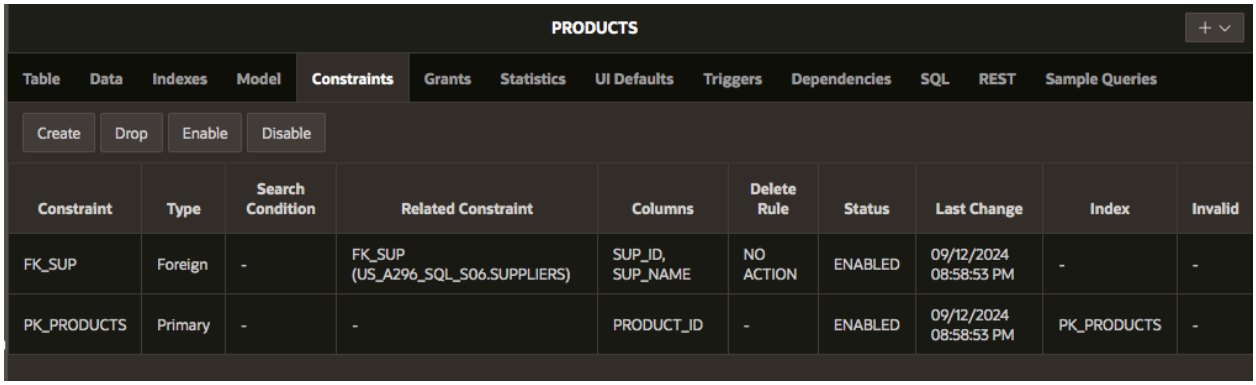



PRODUCTS table

```
CREATE TABLE PRODUCTS (
   product_id NUMBER(10),
   sup_id NUMBER(15),
   sup_name VARCHAR2(30),
   CONSTRAINT pk_products PRIMARY KEY (product_id),
   CONSTRAINT fk_sup FOREIGN KEY (sup_id, sup_name) REFERENCES SUPPLIERS (sup_id,
sup_name)
);
```



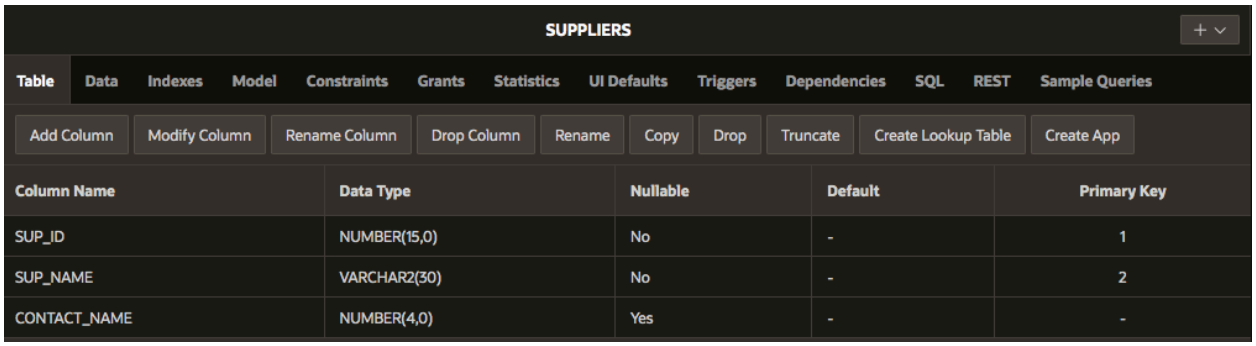



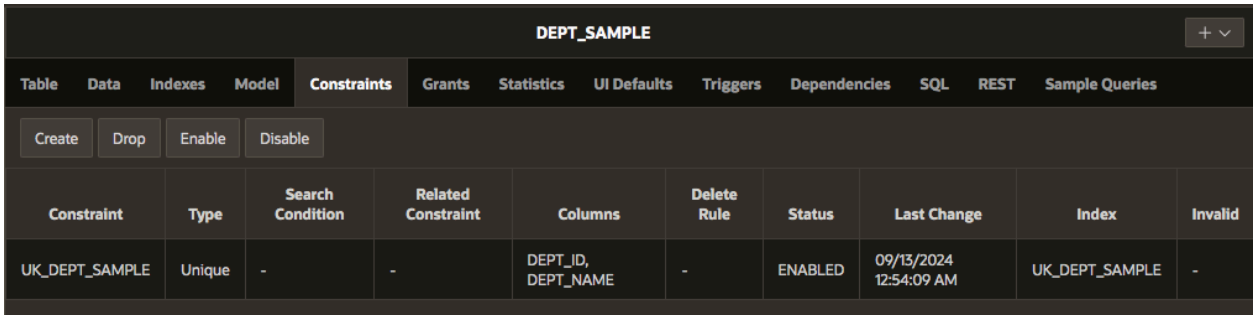

3. Create the DEPT_SAMPLE table with the following structure. The UNIQUE key for this table needs to be defined as a composite comprising of the dept_id and dept_name

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

**ANSWER**:

```
CREATE TABLE DEPT_SAMPLE (
   dept_id NUMBER(8),
   dept_name VARCHAR2(30),
   loc_id NUMBER(4),
   CONSTRAINT uk_dept_sample UNIQUE (dept_id, dept_name)
);
```




**6.4**

**Exercise 1:** Inserting Rows in Tables

1. Click here to see the statements used to insert rows into the tables created for the Academic Database

**Exercise 2:** Updating Rows in the Tables

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

**ANSWER**:

```
ALTER TABLE faculty_login_details
ADD (details VARCHAR2(50))
```

2. Update at least 2 records in the DETAILS column in the faculty login details table. You will have to look up the LOGIN_DATE_TIME values for the records being updated since it is part of the primary key

**ANSWER**:

```
UPDATE faculty_login_details
SET details = 'some random details here'
WHERE login_date_time = '01-JUN-17 05.13.21.000000 PM';

UPDATE faculty_login_details
SET details = 'sea shell sea shell by the sea shore'
WHERE login_date_time = '01-JUN-17 05.13.15.000000 PM';
```

**6.5**

**Exercise 1:** Controlling Transactions

1. Questions are in the image below

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
  ) ;
```

Then the table is altered to add an `email_addr` column:

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

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?

**ANSWER**: The new email field would not be there because the TO SAVEPOINT clause was not mentioned and ROLLBACK itself ends the transaction by discarding all pending data changes (aka rolls back the entire transaction). SAVEPOINT itself does not make any changes permanent, but just saves the current transaction at that point in time.

2. The following steps are done per the image.

```
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;
```

   a. If an INSERT is done to add rows into the test table and a Savepoint is then created called INSERT_DONE.

b. Then an UPDATE to a row in the test table is done and a Savepoint is created called UPDATE_DONE.
c. 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?
    i. **ANSWER**: The records with ID 920, 940, and 960 would remain in the table
d. 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?
    i. **ANSWER**: The transaction will specifically rollback to the UPDATE_DONE savepoint. Anything after this savepoint is discarded, so none of the records would have been deleted.

**6.6**

**Exercise 1:** Retrieving Columns from tables

1. Write a simple query to view the data inserted in the tables created for the academic database
    **a. ANSWER:**
        i. Query Template: SELECT * FROM table_name;
        ii. E.g.
            1. SELECT * FROM students;
            2. SELECT * FROM parent_information;
            3. SELECT * FROM courses;
2. Write a query to retrieve the exam grade obtained by each student for every exam attempted.
    a. SELECT * FROM exam_results ORDER BY student_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.
    a. SELECT student_id, exam_eligibility, work_days_count - days_off_count FROM student_attendances ORDER BY student_id;
4. Display the LOGIN_DATE_TIME for each faculty member.
    a. SELECT * FROM faculty_login_details;
5. Display the name of the Head of the Department for each of the Departments.
    a. SELECT dept_name, head FROM 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
    a. SELECT id || ': FIRST NAME IS ' || first_name FROM students;
7. Display all the distinct exam types from the AD_EXAMS table
    a. SELECT DISTINCT exam_type FROM exams;

**6.7**

**Exercise 1:** Restricting Data Using SELECT

1. Display the course details for the Spring Session.
    a. SELECT * FROM courses WHERE aca_ses_id = 100;
2. Display the details of the students who have scored more than 95.
    a. SELECT * FROM exam_results WHERE grade > 95;
3. Display the details of the students who have scored between 65 and 70.
    a. SELECT * FROM exam_results WHERE grade BETWEEN 65 AND 70;
4. Display the students who registered after 01-Jun-2012.
    a. SELECT * FROM students WHERE registration_yr > '01-Jun-2012';
5. Display the course details for departments 10 and 30.
    a. SELECT * FROM courses WHERE dept_id = 10 OR dept_id = 30;
6. Display the details of students whose first name begins with the letter "J".
    a. SELECT * FROM students WHERE first_name LIKE 'J%';
7. Display the details of students who have opted for courses 190 or 193.
    a. **ANSWER below**:

```
SELECT s.*, scd.*
FROM students s
JOIN student_course_details scd
 ON s.id = scd.student_id
WHERE course_id in (190, 193)
ORDER BY course_id;
```

8. Display the course details offered by department 30 for the Fall Session (Session ID 200).
    a. SELECT * FROM courses WHERE dept_id = 30 AND aca_ses_id = 200;
9. Display the course details of courses not being offered in the summer and fall session (Session ID 200 and 300).
    a. SELECT * FROM courses WHERE aca_ses_id NOT IN (200, 300);
10. Display the course details for department 20
    a. SELECT * FROM courses WHERE dept_id = 20;

**6.8**

**Exercise 1**: Sorting Data Using ORDER BY

1. Display all fields for each of the records in ascending order for the following tables:
    a. AD_STUDENTS ordered by REG_YEAR

**ANSWER:**
SELECT * FROM students ORDER BY registration_yr;

| ID | FIRST_NAME | LAST_NAME | REGISTRATION_YR | EMAIL | PARENT_INFO_ID |
|---|---|---|---|---|---|
| 720 | JACK | SMITH | 01-Jan-2012 | JSMITH@SCHOOL.EDU | 600 |
| 730 | NOAH | AUDRY | 01-Jan-2012 | NAUDRY@SCHOOL.EDU | 640 |
| 760 | JEANNE | BEN | 01-Mar-2012 | JBEN@SCHOOL.EDU | 610 |
| 750 | ROBERT | BEN | 01-Mar-2012 | RBEN@SCHOOL.EDU | 610 |
| 740 | RHONDA | TAYLOR | 01-Sep-2012 | RTAYLOR@SCHOOL.EDU | 620 |
| 770 | MILLS | CARMEN | 01-Apr-2013 | MCARMEN@SCHOOL.EDU | 630 |

b. AD_EXAM_RESULTS ordered by STUDENT_ID and COURSE_ID

**ANSWER:**
SELECT * FROM exam_results ORDER BY student_id, course_id;

| GRADE | STUDENT_ID | EXAM_ID | COURSE_ID |
|---|---|---|---|
| 91 | 720 | 500 | 190 |
| 97 | 720 | 520 | 193 |
| 85 | 730 | 530 | 194 |
| 87 | 730 | 540 | 195 |
| 78 | 750 | 520 | 191 |
| 60 | 750 | 500 | 192 |
| 97 | 750 | 510 | 195 |
| 60 | 760 | 530 | 191 |
| 70 | 760 | 510 | 192 |
| 65 | 760 | 540 | 192 |

10 rows returned in 0.00 seconds Download

c. AD_STUDENT_ATTENDANCE ordered by STUDENT_ID

**ANSWER:**
SELECT * FROM student_attendances ORDER BY student_id;

| WORK_DAYS_COUNT | DAYS_OFF_COUNT | EXAM_ELIGIBILITY | STUDENT_ID | ACADEMIC_SESSION_ID |
|---|---|---|---|---|
| 180 | 21 | Y | 720 | 100 |
| 180 | 11 | Y | 730 | 200 |
| 180 | 12 | Y | 740 | 300 |
| 180 | 14 | Y | 750 | 100 |
| 180 | 15 | Y | 760 | 200 |
| 180 | 13 | Y | 770 | 300 |

d. AD_DEPARTMENTS ordered by the department ID

**ANSWER:**
SELECT * FROM departments ORDER BY id;

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

4 rows returned in 0.00 seconds Download

2. Display the percentage of days students have taken days off and sort the records based on the percentage calculated.
    a. **ANSWER below:**

```
SELECT
   student_id,
   ROUND(100 * (days_off_count / (work_days_count + days_off_count)), 2)
percentage_of_days_off
FROM student_attendances
ORDER BY percentage_of_days_off;
```

| STUDENT_ID | PERCENTAGE_OF_DAYS_OFF |
|---|---|
| 730 | 5.76 |
| 740 | 6.25 |
| 770 | 6.74 |
| 750 | 7.22 |
| 760 | 7.69 |
| 720 | 10.45 |

3. Display the top 5 students based on exam grade results.
    a. **ANSWER below:**

```
SELECT ROWNUM AS student_id, grade
FROM
   (SELECT student_id, grade
   FROM exam_results
   ORDER BY grade DESC)
WHERE ROWNUM <=5;
```

| STUDENT_ID | GRADE |
|---|---|
| 1 | 97 |
| 2 | 97 |
| 3 | 91 |
| 4 | 87 |
| 5 | 85 |

5 rows returned in 0.01 seconds Download

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

SELECT * FROM parent_information ORDER BY id;

| ID | PARENT1_FIRST_NAME | PARENT1_LAST_NAME | PARENT2_FIRST_NAME | PARENT2_LAST_NAME |
|---|---|---|---|---|
| 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 |

5 rows returned in 0.01 seconds Download

**6.9**

**Exercise 1:** Using JOINS in SQL Queries

1. Display the different courses offered by the departments in the school.
    a. **ANSWER below:**

SELECT d.dept_name, c.course_name
FROM courses c
JOIN departments d
ON (c.dept_id = d.id)
ORDER BY d.dept_name, c.course_name;

| DEPT_NAME | COURSE_NAME |
|---|---|
| ACCOUNTING | COST ACCOUNTING |
| ACCOUNTING | INTRODUCTION TO BUSINESS LAW |
| ACCOUNTING | PRINCIPLES OF ACCOUNTING |
| ACCOUNTING | STRATEGIC TAX PLANNING FOR BUSINESS |
| BIOLOGY | CELL BIOLOGY |
| BIOLOGY | GENERAL BIOLOGY |

2. Display the courses offered in the Fall session.
    a. **ANSWER below:**

```
SELECT s.session_name, c.course_name
FROM academic_sessions s
JOIN courses c
ON (s.id = 200)
AND c.aca_ses_id = 200;
```

| SESSION_NAME | COURSE_NAME |
|---|---|
| FALL SESSION | CELL BIOLOGY |
| FALL SESSION | GENERAL BIOLOGY |

3. Display the course details, the department that offers the courses and students who have enrolled for those courses.
    a. **ANSWER below:**

```
SELECT s.first_name || ' ' || s.last_name AS name, c.course_name, d.dept_name
FROM students s
JOIN student_course_details scd
ON (s.id = scd.student_id)
JOIN courses c
ON (scd.course_id = c.id)
JOIN departments d
ON (c.dept_id = d.id);
```

| NAME | COURSE_NAME | DEPT_NAME |
|---|---|---|
| JEANNE BEN | PRINCIPLES OF ACCOUNTING | ACCOUNTING |
| JACK SMITH | PRINCIPLES OF ACCOUNTING | ACCOUNTING |
| NOAH AUDRY | INTRODUCTION TO BUSINESS LAW | ACCOUNTING |
| JEANNE BEN | COST ACCOUNTING | ACCOUNTING |
| ROBERT BEN | COST ACCOUNTING | ACCOUNTING |
| MILLS CARMEN | COST ACCOUNTING | ACCOUNTING |
| MILLS CARMEN | STRATEGIC TAX PLANNING FOR BUSINESS | ACCOUNTING |
| JACK SMITH | STRATEGIC TAX PLANNING FOR BUSINESS | ACCOUNTING |
| MILLS CARMEN | GENERAL BIOLOGY | BIOLOGY |
| RHONDA TAYLOR | CELL BIOLOGY | BIOLOGY |

4. Display the course details, the department that offers the courses and students who have enrolled for those courses for department 20.
    a. **ANSWER below:**

```
SELECT s.first_name || ' ' || s.last_name AS name, c.course_name, d.dept_name
FROM students s
JOIN student_course_details scd
ON (s.id = scd.student_id)
JOIN courses c
ON (scd.course_id = c.id)
JOIN departments d
ON (c.dept_id = d.id)
AND (c.dept_id = 20);
```

| NAME | COURSE_NAME | DEPT_NAME |
|---|---|---|
| MILLS CARMEN | GENERAL BIOLOGY | BIOLOGY |
| RHONDA TAYLOR | CELL BIOLOGY | BIOLOGY |

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.
    a. **ANSWER below:**

```
SELECT sdt.id, sdt.first_name || ' ' || sdt.last_name AS name, crse.course_name, res.grade
FROM student_course_details scd
JOIN courses crse
ON (scd.course_id = crse.id)
JOIN exam_results res
ON (crse.id = res.course_id)
AND crse.id BETWEEN 190 AND 192
JOIN students sdt
ON (sdt.id = scd.student_id)
ORDER BY crse.course_name, name;
```

| ID | NAME | COURSE_NAME | GRADE |
|---|---|---|---|
| 760 | JEANNE BEN | COST ACCOUNTING | 60 |
| 760 | JEANNE BEN | COST ACCOUNTING | 65 |
| 760 | JEANNE BEN | COST ACCOUNTING | 70 |
| 770 | MILLS CARMEN | COST ACCOUNTING | 60 |
| 770 | MILLS CARMEN | COST ACCOUNTING | 65 |
| 770 | MILLS CARMEN | COST ACCOUNTING | 70 |
| 750 | ROBERT BEN | COST ACCOUNTING | 70 |
| 750 | ROBERT BEN | COST ACCOUNTING | 60 |
| 750 | ROBERT BEN | COST ACCOUNTING | 65 |
| 730 | NOAH AUDRY | INTRODUCTION TO BUSINESS LAW | 78 |
| 730 | NOAH AUDRY | INTRODUCTION TO BUSINESS LAW | 60 |
| 720 | JACK SMITH | PRINCIPLES OF ACCOUNTING | 91 |
| 760 | JEANNE BEN | PRINCIPLES OF ACCOUNTING | 91 |

13 rows returned in 0.00 seconds Download

6. Retrieve the rows from the AD_EXAM_RESULTS table even if there are no matching records in the AD_COURSES table.
    a. **ANSWER below:**

```
SELECT *
FROM exam_results res FULL OUTER JOIN courses crse
ON (res.course_id = crse.id)
ORDER BY crse.id;
```

| GRADE | STUDENT_ID | EXAM_ID | COURSE_ID | ID | COURSE_NAME | ACA_SES_ID | DEPT_ID | LOGIN_ID | BUILDING | ROOM | DATE_TIME |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 91 | 720 | 500 | 190 | 190 | PRINCIPLES OF ACCOUNTING | 100 | 10 | - | A | 101 | MWF 12-1 |
| 78 | 750 | 520 | 191 | 191 | INTRODUCTION TO BUSINESS LAW | 100 | 10 | - | B | 201 | THUR 2-4 |
| 60 | 760 | 530 | 191 | 191 | INTRODUCTION TO BUSINESS LAW | 100 | 10 | - | B | 201 | THUR 2-4 |
| 65 | 760 | 540 | 192 | 192 | COST ACCOUNTING | 100 | 10 | - | C | 301 | TUES 5-7 |
| 70 | 760 | 510 | 192 | 192 | COST ACCOUNTING | 100 | 10 | - | C | 301 | TUES 5-7 |
| 60 | 750 | 500 | 192 | 192 | COST ACCOUNTING | 100 | 10 | - | C | 301 | TUES 5-7 |
| 97 | 720 | 520 | 193 | 193 | STRATEGIC TAX PLANNING FOR BUSINESS | 100 | 10 | TAX123 | - | - | - |
| 85 | 730 | 530 | 194 | 194 | GENERAL BIOLOGY | 200 | 20 | BIO123 | - | - | - |
| 97 | 750 | 510 | 195 | 195 | CELL BIOLOGY | 200 | 20 | - | D | 401 | MWF 9-10 |
| 87 | 730 | 540 | 195 | 195 | CELL BIOLOGY | 200 | 20 | - | D | 401 | MWF 9-10 |

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

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

**ANSWER**: It would create a Cartesian product.

| ID | DATE_START | EXAM_TYPE | COURSE_ID | EXAM_TYPE | EXAM_NAME | EXAM_DESCRIPTION |
|---|---|---|---|---|---|---|
| 500 | 12-Sep-2013 | MCE | 190 | MCE | Multiple Choice Exams | CHOOSE MORE THAN ONE ANSWER |
| 500 | 12-Sep-2013 | MCE | 190 | TF | TRUE AND FALSE Exams | CHOOSE EITHER TRUE OR FALSE |
| 500 | 12-Sep-2013 | MCE | 190 | ESS | ESSAY Exams | WRITE PARAGRAPHS |
| 500 | 12-Sep-2013 | MCE | 190 | SA | SHORT ANSWER Exams | WRITE SHORT ANSWERS |
| 500 | 12-Sep-2013 | MCE | 190 | FIB | FILL IN THE BLANKS Exams | TYPE IN THE CORRECT ANSWER |
| 510 | 15-Sep-2013 | SA | 191 | MCE | Multiple Choice Exams | CHOOSE MORE THAN ONE ANSWER |
| 510 | 15-Sep-2013 | SA | 191 | TF | TRUE AND FALSE Exams | CHOOSE EITHER TRUE OR FALSE |
| 510 | 15-Sep-2013 | SA | 191 | ESS | ESSAY Exams | WRITE PARAGRAPHS |
| 510 | 15-Sep-2013 | SA | 191 | SA | SHORT ANSWER Exams | WRITE SHORT ANSWERS |
| 510 | 15-Sep-2013 | SA | 191 | FIB | FILL IN THE BLANKS Exams | TYPE IN THE CORRECT ANSWER |
| 520 | 18-Sep-2013 | FIB | 192 | MCE | Multiple Choice Exams | CHOOSE MORE THAN ONE ANSWER |
| 520 | 18-Sep-2013 | FIB | 192 | TF | TRUE AND FALSE Exams | CHOOSE EITHER TRUE OR FALSE |
| 520 | 18-Sep-2013 | FIB | 192 | ESS | ESSAY Exams | WRITE PARAGRAPHS |
| 520 | 18-Sep-2013 | FIB | 192 | SA | SHORT ANSWER Exams | WRITE SHORT ANSWERS |
| 520 | 18-Sep-2013 | FIB | 192 | FIB | FILL IN THE BLANKS Exams | TYPE IN THE CORRECT ANSWER |

More than 15 rows available. Increase rows selector to view more rows.