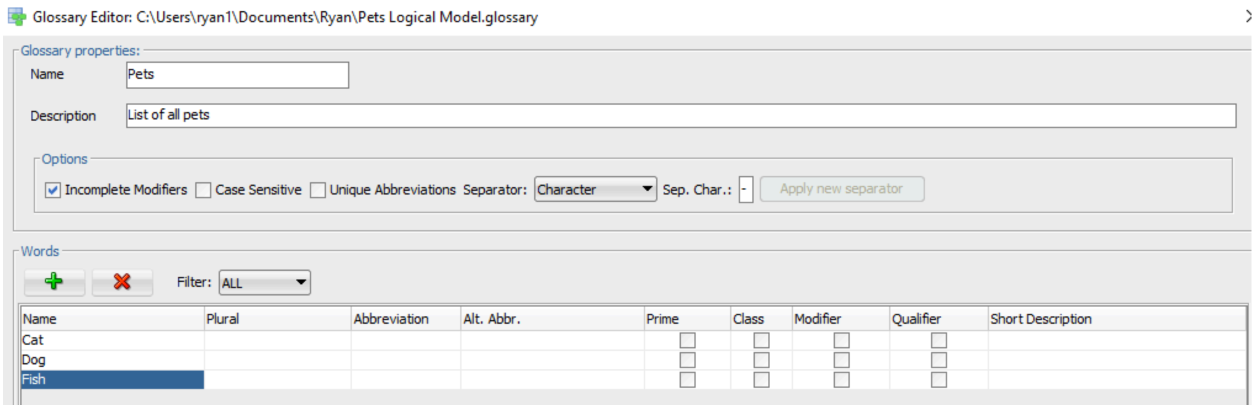
5-1: Mapping Entities and Attributes


Glossary Editor: C:\Users\ryan1\Documents\Ryan\Pets Logical Model.glossary

Glossary properties:

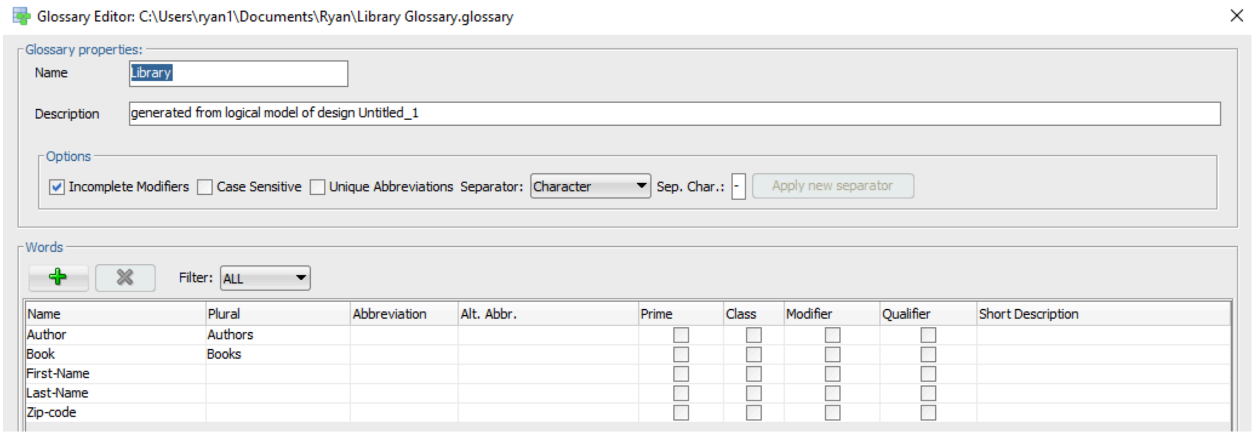
Name: Pets

Description: List of all pets

Options: [✓] Incomplete Modifiers [ ] Case Sensitive [ ] Unique Abbreviations Separator: Character Sep. Char.: -

Words — Filter: ALL

Name	Plural	Abbreviation	Alt. Abbr.	Prime	Class	Modifier	Qualifier	Short Description
Cat				☐	☐	☐	☐	
Dog				☐	☐	☐	☐	
Fish				☐	☐	☐	☐	


Glossary Editor: C:\Users\ryan1\Documents\Ryan\Library Glossary.glossary

Glossary properties:

Name: Library

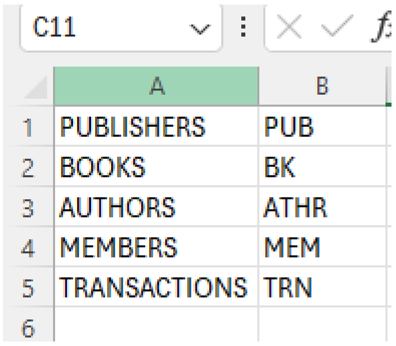
Description: generated from logical model of design Untitled_1

Options: [✓] Incomplete Modifiers [ ] Case Sensitive [ ] Unique Abbreviations Separator: Character Sep. Char.: -

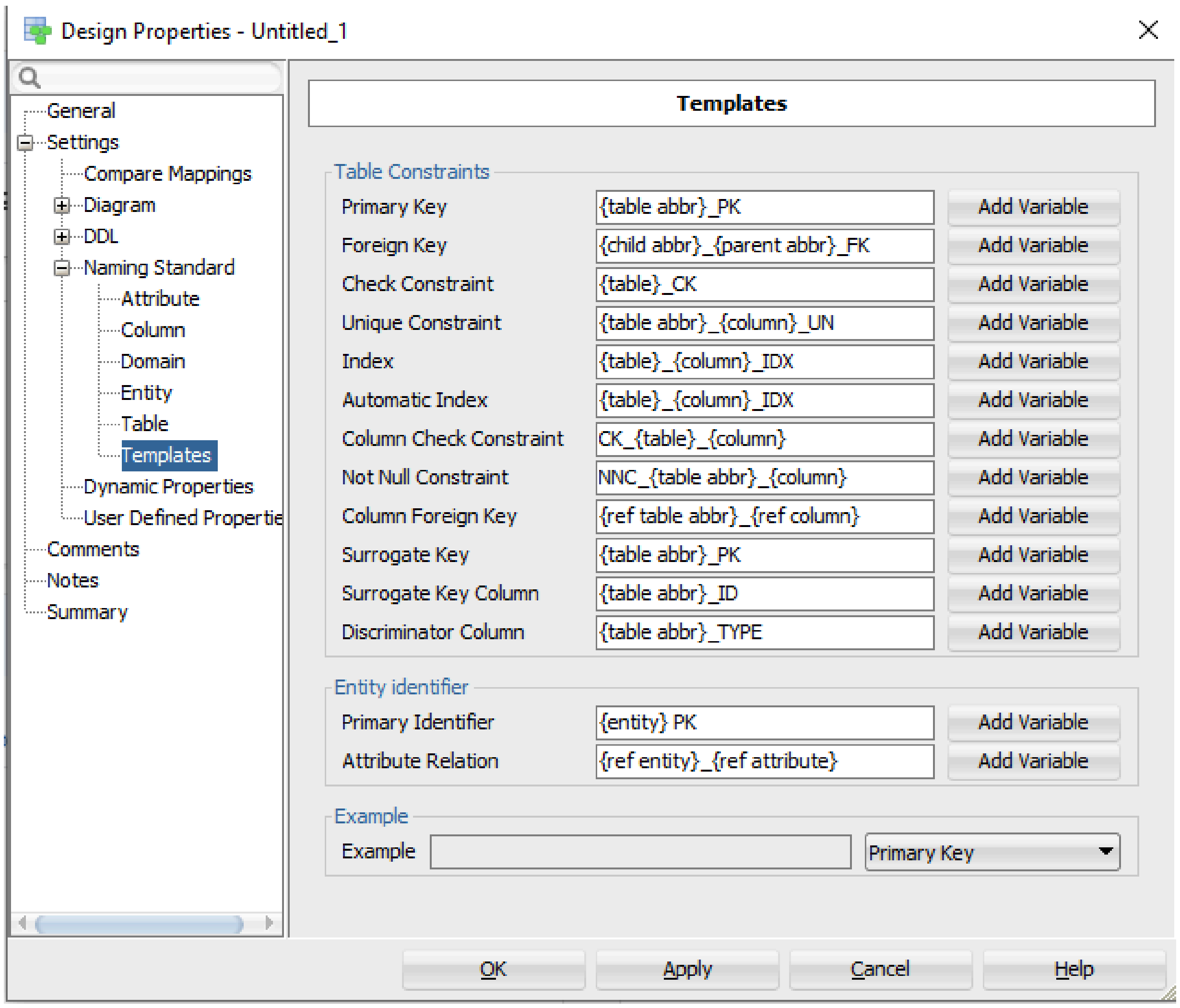
Words — Filter: ALL

Name	Plural	Abbreviation	Alt. Abbr.	Prime	Class	Modifier	Qualifier	Short Description
Author	Authors			☐	☐	☐	☐	
Book	Books			☐	☐	☐	☐	
First-Name				☐	☐	☐	☐	
Last-Name				☐	☐	☐	☐	
Zip-code				☐	☐	☐	☐	

5-2: Mapping Primary and Foreign Keys Practice


C11

	A	B
1	PUBLISHERS	PUB
2	BOOKS	BK
3	AUTHORS	ATHR
4	MEMBERS	MEM
5	TRANSACTIONS	TRN
6		



Templates

Table Constraints

Primary Key	{table abbr}_PK	Add Variable
Foreign Key	{child abbr}_{parent abbr}_FK	Add Variable
Check Constraint	{table}_CK	Add Variable
Unique Constraint	{table abbr}_{column}_UN	Add Variable
Index	{table}_{column}_IDX	Add Variable
Automatic Index	{table}_{column}_IDX	Add Variable
Column Check Constraint	CK_{table}_{column}	Add Variable
Not Null Constraint	NNC_{table abbr}_{column}	Add Variable
Column Foreign Key	{ref table abbr}_{ref column}	Add Variable
Surrogate Key	{table abbr}_PK	Add Variable
Surrogate Key Column	{table abbr}_ID	Add Variable
Discriminator Column	{table abbr}_TYPE	Add Variable

Entity identifier

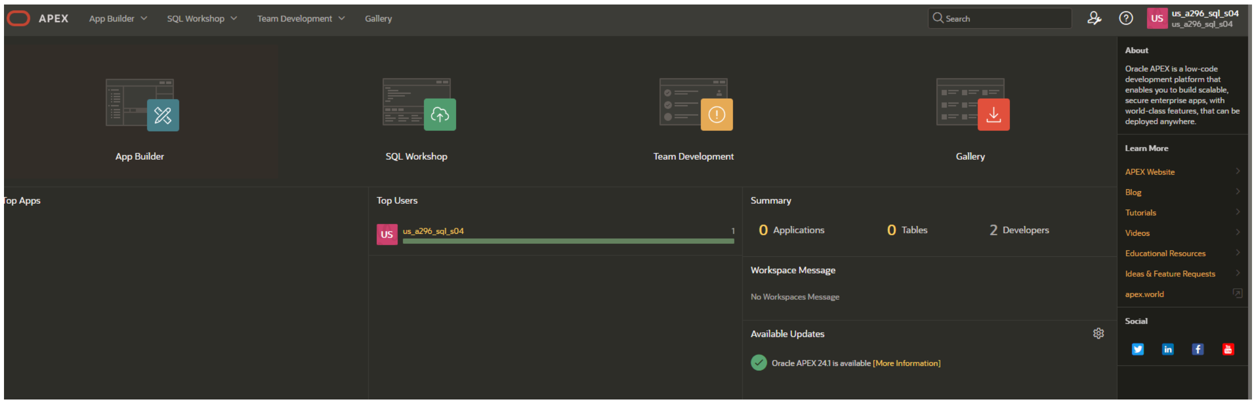
Primary Identifier	{entity} PK	Add Variable
Attribute Relation	{ref entity}_{ref attribute}	Add Variable

Example

Example: Primary Key

```
Oracle SQL Developer Data Modeler Names Abbreviations Log.
Date and Time: 2024-09-17 21:54:08 EDT
Design Name: Untitled_1

          Standardized Objects:
                    Tables:    0
                    Columns:   0
                    Indexes:   0
                    Views:     0
```

6-1 : Introduction to Oracle Application Express Practices



About

Oracle APEX is a low-code development platform that enables you to build scalable, secure enterprise apps, with world-class features, that can be deployed anywhere.

Top Users: us_a296_sql_s04 1

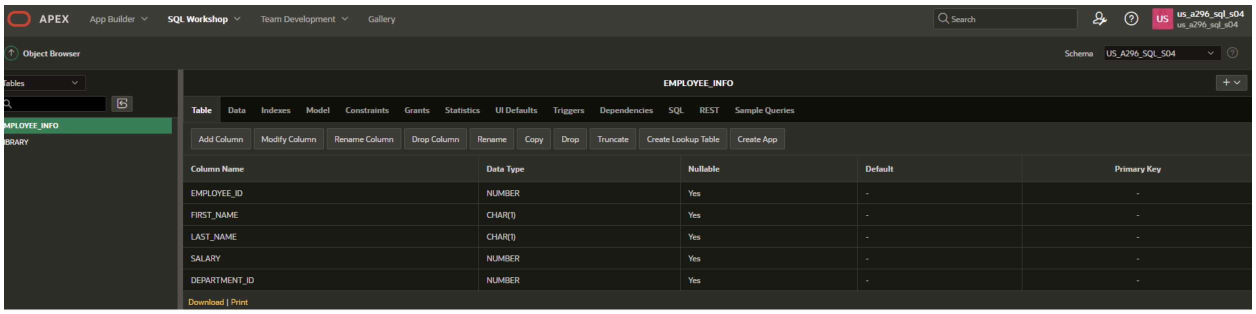
Summary: 0 Applications 0 Tables 2 Developers

Workspace Message

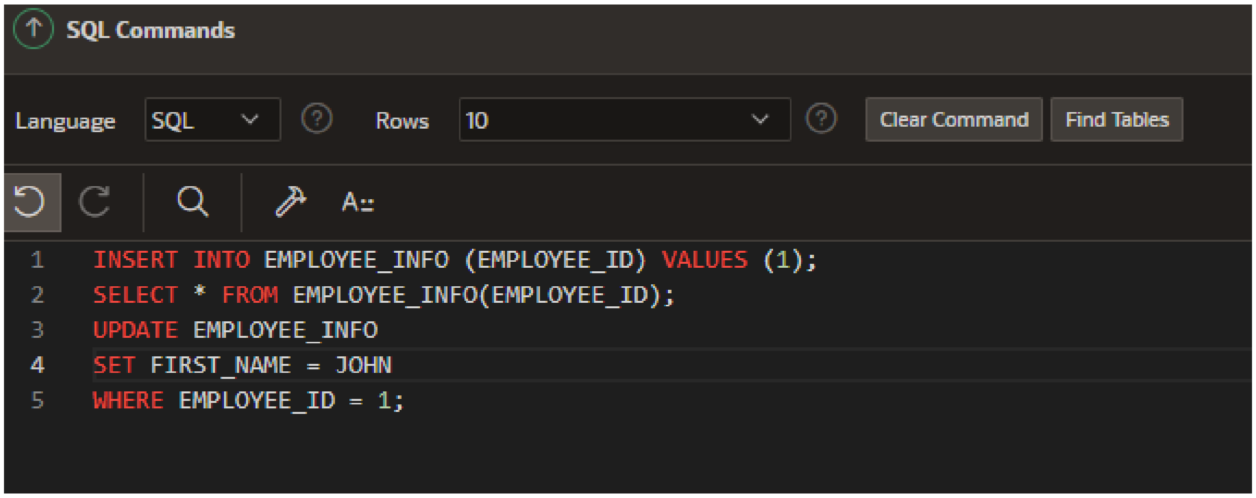
No Workspaces Message

Available Updates

Oracle APEX 24.1 is available [More Information]



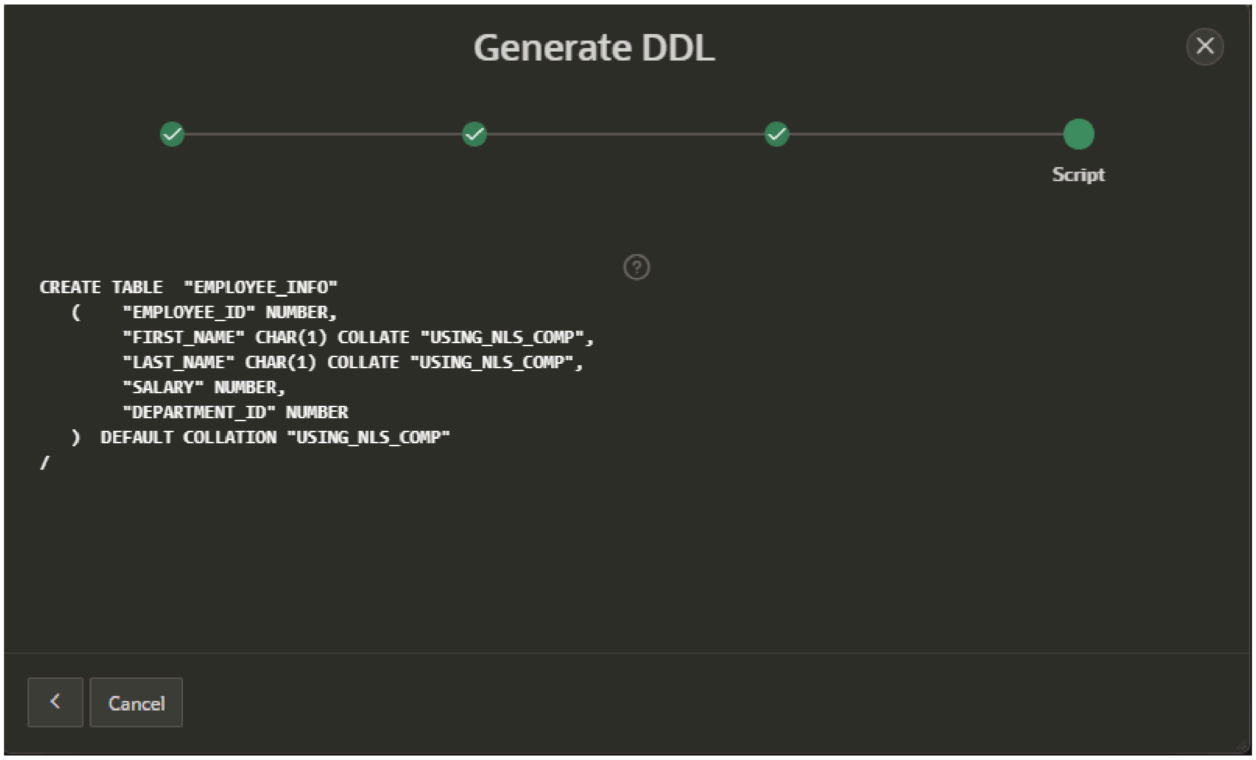
Column Name	Data Type	Nullable	Default	Primary Key
EMPLOYEE_ID	NUMBER	Yes	-	-
FIRST_NAME	CHAR(1)	Yes	-	-
LAST_NAME	CHAR(1)	Yes	-	-
SALARY	NUMBER	Yes	-	-
DEPARTMENT_ID	NUMBER	Yes	-	-



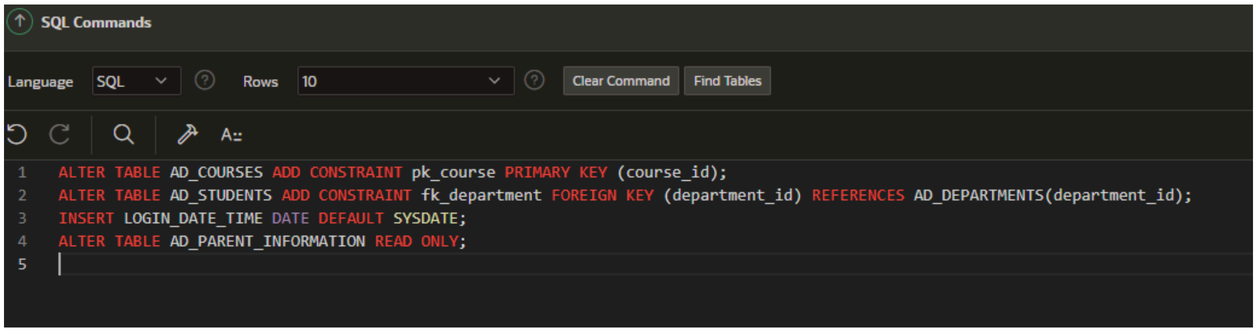
Language: SQL Rows: 10

```
INSERT INTO EMPLOYEE_INFO (EMPLOYEE_ID) VALUES (1);
SELECT * FROM EMPLOYEE_INFO(EMPLOYEE_ID);
UPDATE EMPLOYEE_INFO
SET FIRST_NAME = JOHN
WHERE EMPLOYEE_ID = 1;
```

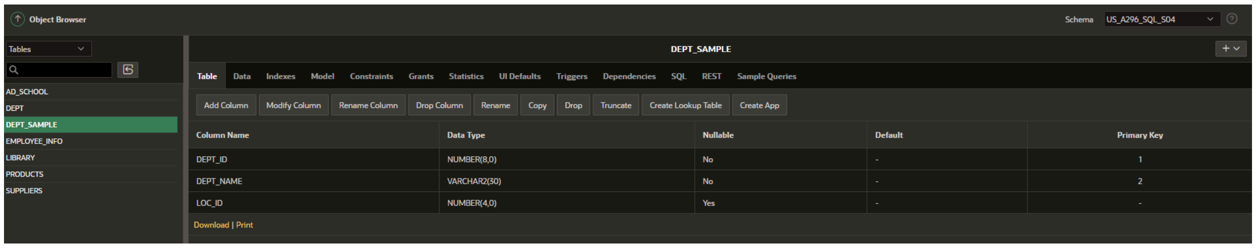
6-3 : Defining Data Definition Language (DDL) Practices



```
CREATE TABLE  "EMPLOYEE_INFO"
   (    "EMPLOYEE_ID" NUMBER,
        "FIRST_NAME" CHAR(1) COLLATE "USING_NLS_COMP",
        "LAST_NAME" CHAR(1) COLLATE "USING_NLS_COMP",
        "SALARY" NUMBER,
        "DEPARTMENT_ID" NUMBER
   ) DEFAULT COLLATION "USING_NLS_COMP"
/
```

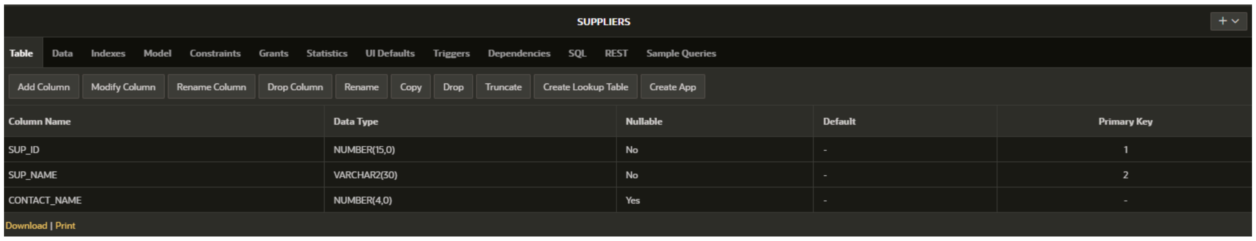


```
ALTER TABLE AD_COURSES ADD CONSTRAINT pk_course PRIMARY KEY (course_id);
ALTER TABLE AD_STUDENTS ADD CONSTRAINT fk_department FOREIGN KEY (department_id) REFERENCES AD_DEPARTMENTS(department_id);
INSERT LOGIN_DATE_TIME DATE DEFAULT SYSDATE;
ALTER TABLE AD_PARENT_INFORMATION READ ONLY;
```



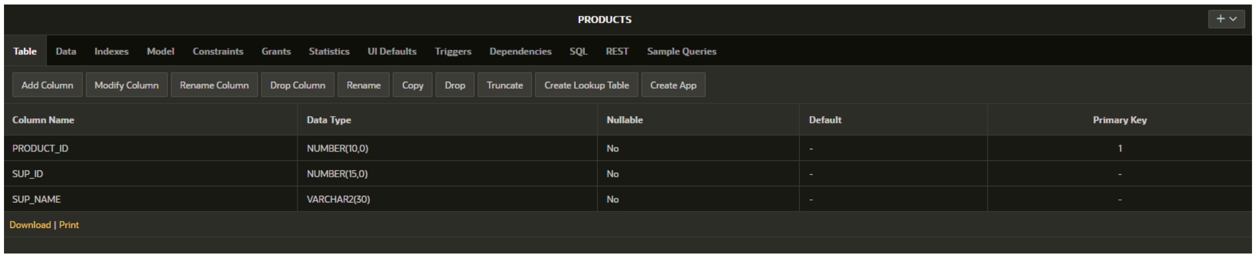
DEPT_SAMPLE

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



SUPPLIERS

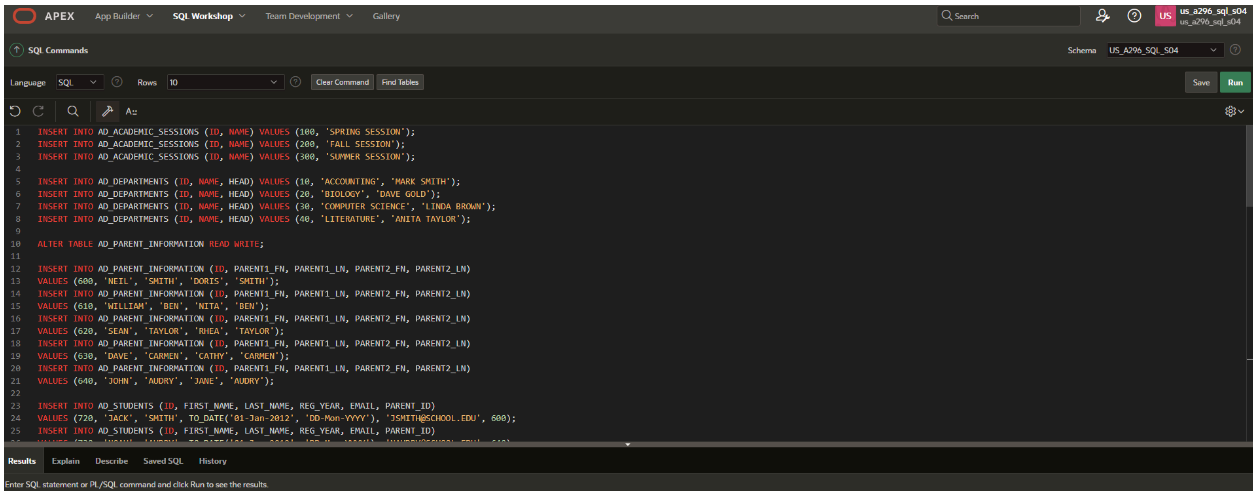
Column Name	Data Type	Nullable	Default	Primary Key
SUP_ID	NUMBER(15,0)	No	-	1
SUP_NAME	VARCHAR2(30)	No	-	2
CONTACT_NAME	NUMBER(4,0)	Yes	-	-



PRODUCTS

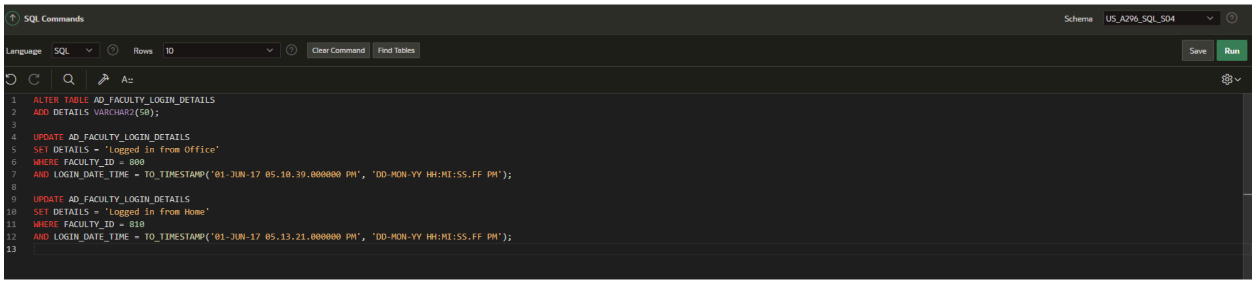
Column Name	Data Type	Nullable	Default	Primary Key
PRODUCT_ID	NUMBER(10,0)	No	-	1
SUP_ID	NUMBER(15,0)	No	-	-
SUP_NAME	VARCHAR2(30)	No	-	-

6-4 : Defining Data Manipulation Practices



```sql
INSERT INTO AD_ACADEMIC_SESSIONS (ID, NAME) VALUES (100, 'SPRING SESSION');
INSERT INTO AD_ACADEMIC_SESSIONS (ID, NAME) VALUES (200, 'FALL SESSION');
INSERT INTO AD_ACADEMIC_SESSIONS (ID, NAME) VALUES (300, 'SUMMER SESSION');

INSERT INTO AD_DEPARTMENTS (ID, NAME, HEAD) VALUES (10, 'ACCOUNTING', 'MARK SMITH');
INSERT INTO AD_DEPARTMENTS (ID, NAME, HEAD) VALUES (20, 'BIOLOGY', 'DAVE GOLD');
INSERT INTO AD_DEPARTMENTS (ID, NAME, HEAD) VALUES (30, 'COMPUTER SCIENCE', 'LINDA BROWN');
INSERT INTO AD_DEPARTMENTS (ID, NAME, HEAD) VALUES (40, 'LITERATURE', 'ANITA TAYLOR');

ALTER TABLE AD_PARENT_INFORMATION READ WRITE;

INSERT INTO AD_PARENT_INFORMATION (ID, PARENT1_FN, PARENT1_LN, PARENT2_FN, PARENT2_LN)
VALUES (600, 'NEIL', 'SMITH', 'DORIS', 'SMITH');
INSERT INTO AD_PARENT_INFORMATION (ID, PARENT1_FN, PARENT1_LN, PARENT2_FN, PARENT2_LN)
VALUES (610, 'WILLIAM', 'BEN', 'NITA', 'BEN');
INSERT INTO AD_PARENT_INFORMATION (ID, PARENT1_FN, PARENT1_LN, PARENT2_FN, PARENT2_LN)
VALUES (620, 'SEAN', 'TAYLOR', 'RHEA', 'TAYLOR');
INSERT INTO AD_PARENT_INFORMATION (ID, PARENT1_FN, PARENT1_LN, PARENT2_FN, PARENT2_LN)
VALUES (630, 'DAVE', 'CARMEN', 'CATHY', 'CARMEN');
INSERT INTO AD_PARENT_INFORMATION (ID, PARENT1_FN, PARENT1_LN, PARENT2_FN, PARENT2_LN)
VALUES (640, 'JOHN', 'AUDRY', 'JANE', 'AUDRY');

INSERT INTO AD_STUDENTS (ID, FIRST_NAME, LAST_NAME, REG_YEAR, EMAIL, PARENT_ID)
VALUES (720, 'JACK', 'SMITH', TO_DATE('01-Jan-2012', 'DD-Mon-YYYY'), 'JSMITH@SCHOOL.EDU', 600);
INSERT INTO AD_STUDENTS (ID, FIRST_NAME, LAST_NAME, REG_YEAR, EMAIL, PARENT_ID)
```

Enter SQL statement or PL/SQL command and click Run to see the results.



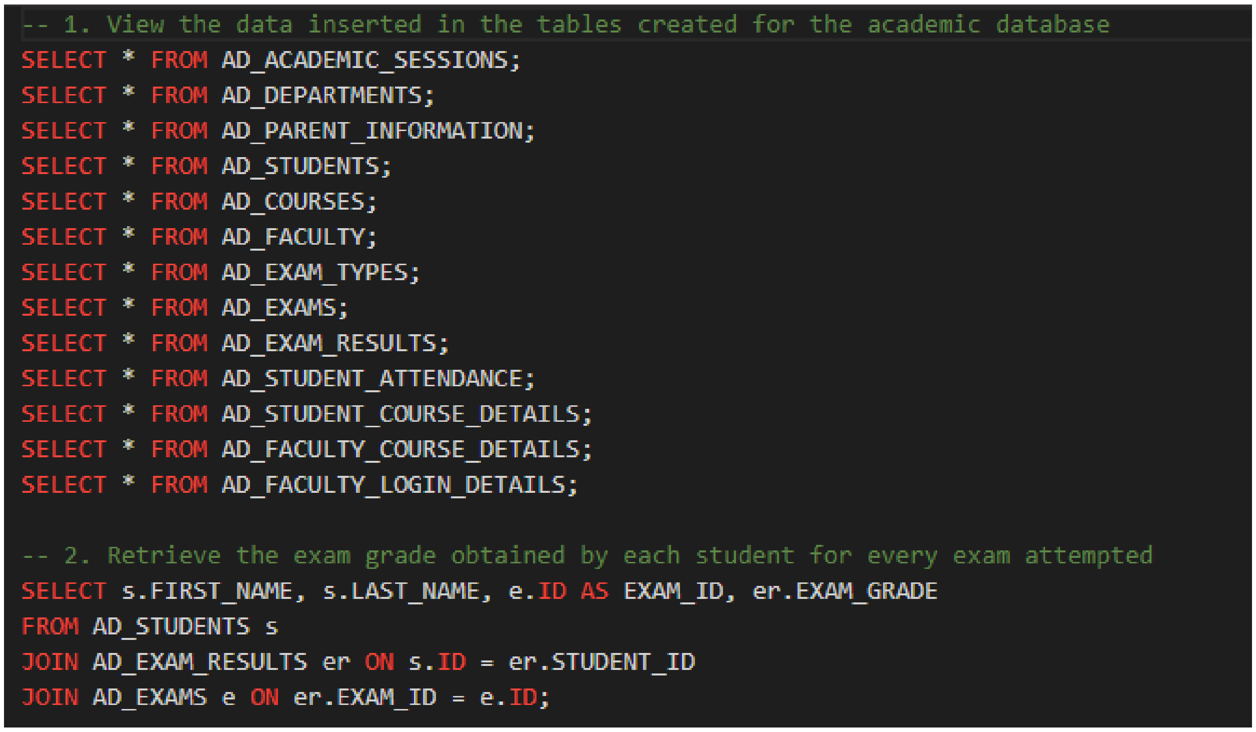
```sql
ALTER TABLE AD_FACULTY_LOGIN_DETAILS
ADD DETAILS VARCHAR2(50);

UPDATE AD_FACULTY_LOGIN_DETAILS
SET DETAILS = 'Logged in from Office'
WHERE FACULTY_ID = 800
AND LOGIN_DATE_TIME = TO_TIMESTAMP('01-JUN-17 05.10.39.000000 PM', 'DD-MON-YY HH:MI:SS.FF PM');

UPDATE AD_FACULTY_LOGIN_DETAILS
SET DETAILS = 'Logged in from Home'
WHERE FACULTY_ID = 810
AND LOGIN_DATE_TIME = TO_TIMESTAMP('01-JUN-17 05.13.21.000000 PM', 'DD-MON-YY HH:MI:SS.FF PM');
```

6-5: Defining Transaction Control Practices

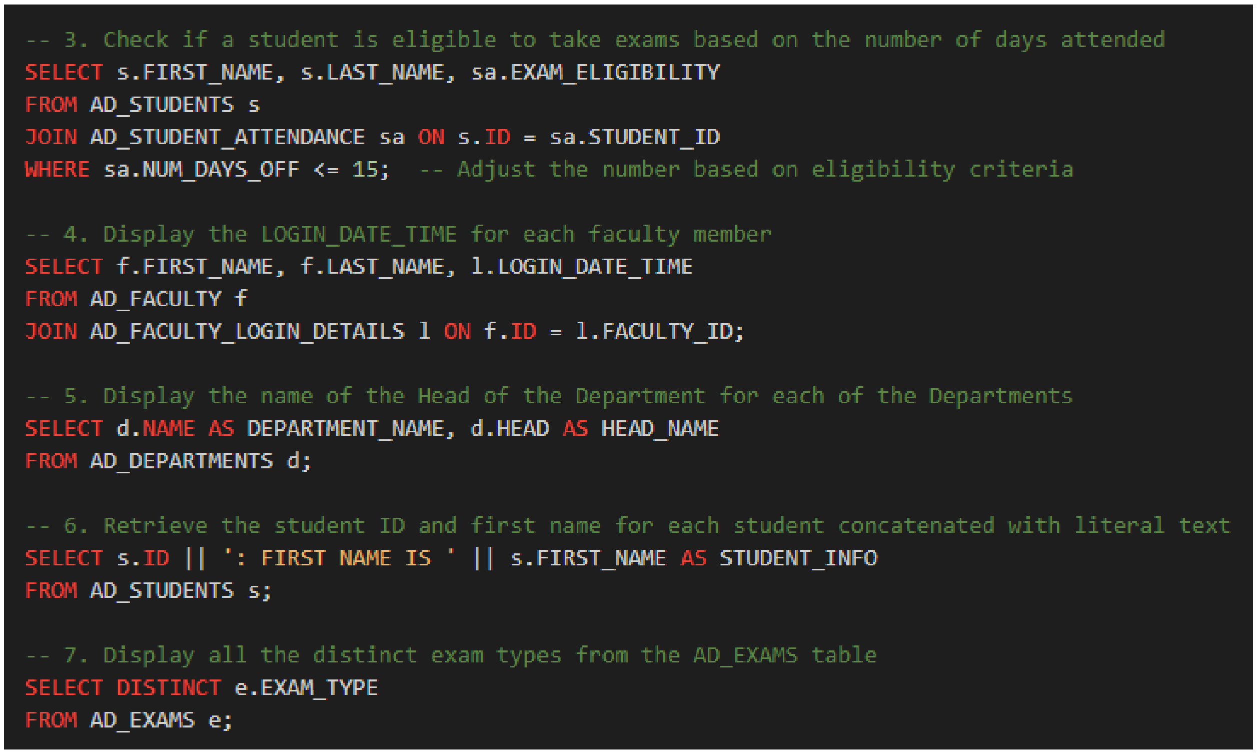
1. If you roll back to the savepoint created after adding the EMAIL_ADDR column, the column will be removed. Rolling back undoes any changes made after that savepoint, so the table will revert to its original state before the column was added.
2. After rolling back to the UPDATE_DONE savepoint, the table will have the rows that were inserted before INSERT_DONE, with any updates applied by the UPDATE operation still in place. However, any rows deleted after UPDATE_DONE will be restored

because the DELETE operation will be undone. So, the table will reflect the state of the data right after the update but before any deletions.

6-6: Retrieving Data Practices

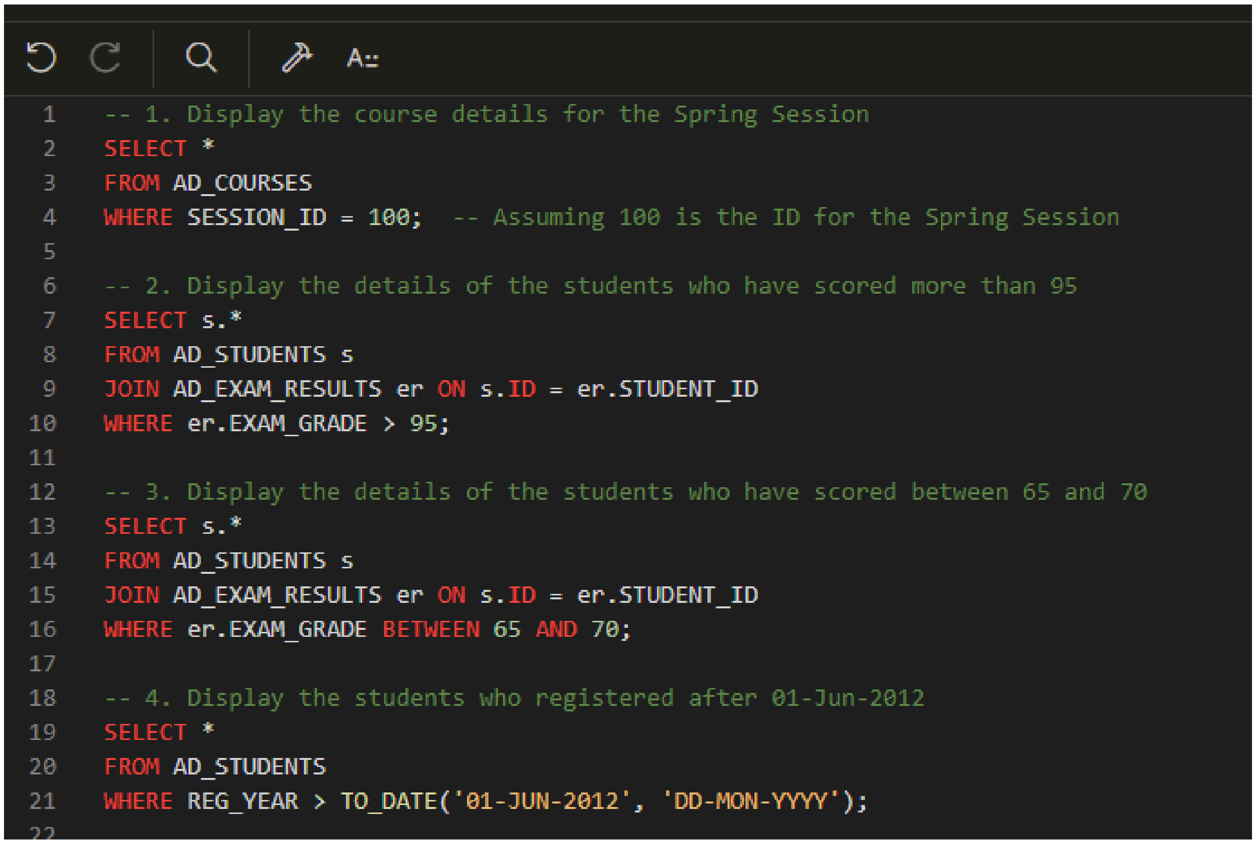


```sql
-- 1. View the data inserted in the tables created for the academic database
SELECT * FROM AD_ACADEMIC_SESSIONS;
SELECT * FROM AD_DEPARTMENTS;
SELECT * FROM AD_PARENT_INFORMATION;
SELECT * FROM AD_STUDENTS;
SELECT * FROM AD_COURSES;
SELECT * FROM AD_FACULTY;
SELECT * FROM AD_EXAM_TYPES;
SELECT * FROM AD_EXAMS;
SELECT * FROM AD_EXAM_RESULTS;
SELECT * FROM AD_STUDENT_ATTENDANCE;
SELECT * FROM AD_STUDENT_COURSE_DETAILS;
SELECT * FROM AD_FACULTY_COURSE_DETAILS;
SELECT * FROM AD_FACULTY_LOGIN_DETAILS;

-- 2. Retrieve the exam grade obtained by each student for every exam attempted
SELECT s.FIRST_NAME, s.LAST_NAME, e.ID AS EXAM_ID, er.EXAM_GRADE
FROM AD_STUDENTS s
JOIN AD_EXAM_RESULTS er ON s.ID = er.STUDENT_ID
JOIN AD_EXAMS e ON er.EXAM_ID = e.ID;
```

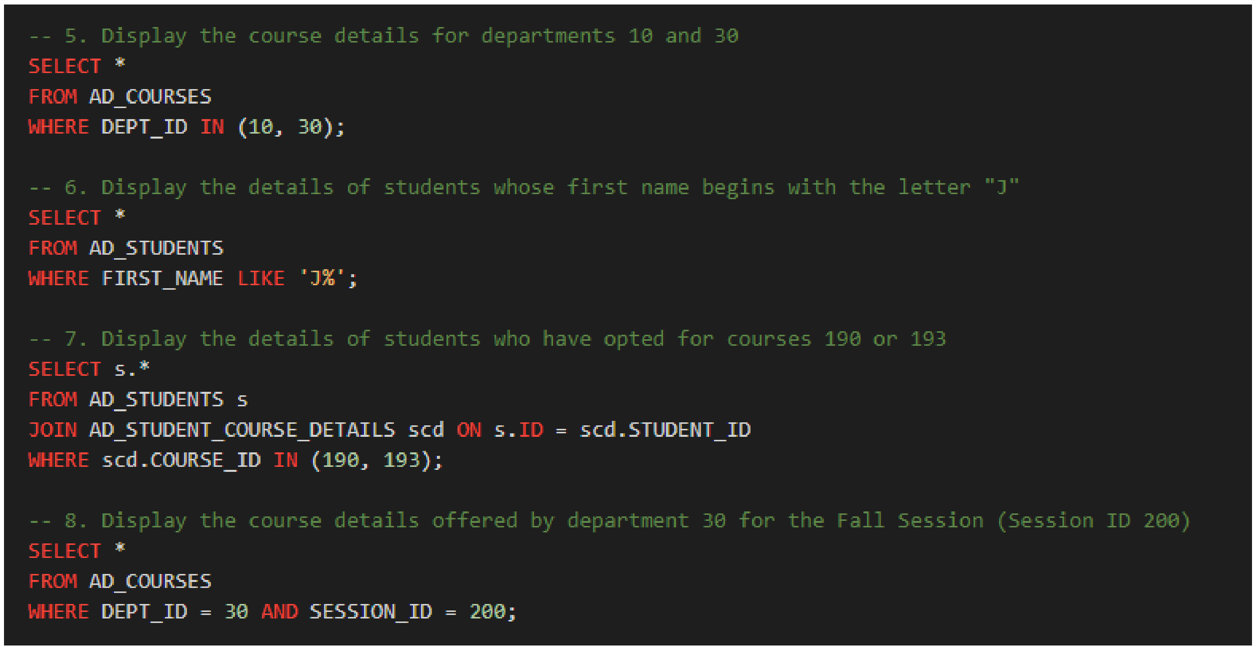


```sql
-- 3. Check if a student is eligible to take exams based on the number of days attended
SELECT s.FIRST_NAME, s.LAST_NAME, sa.EXAM_ELIGIBILITY
FROM AD_STUDENTS s
JOIN AD_STUDENT_ATTENDANCE sa ON s.ID = sa.STUDENT_ID
WHERE sa.NUM_DAYS_OFF <= 15;  -- Adjust the number based on eligibility criteria

-- 4. Display the LOGIN_DATE_TIME for each faculty member
SELECT f.FIRST_NAME, f.LAST_NAME, l.LOGIN_DATE_TIME
FROM AD_FACULTY f
JOIN AD_FACULTY_LOGIN_DETAILS l ON f.ID = l.FACULTY_ID;

-- 5. Display the name of the Head of the Department for each of the Departments
SELECT d.NAME AS DEPARTMENT_NAME, d.HEAD AS HEAD_NAME
FROM AD_DEPARTMENTS d;

-- 6. Retrieve the student ID and first name for each student concatenated with literal text
SELECT s.ID || ': FIRST NAME IS ' || s.FIRST_NAME AS STUDENT_INFO
FROM AD_STUDENTS s;

-- 7. Display all the distinct exam types from the AD_EXAMS table
SELECT DISTINCT e.EXAM_TYPE
FROM AD_EXAMS e;
```

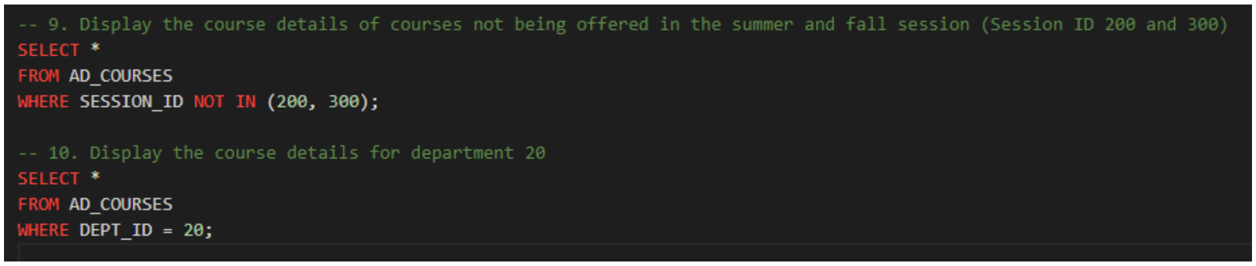
6-7: Restricting Data Using WHERE Statement Practices



```sql
-- 1. Display the course details for the Spring Session
SELECT *
FROM AD_COURSES
WHERE SESSION_ID = 100;  -- Assuming 100 is the ID for the Spring Session

-- 2. Display the details of the students who have scored more than 95
SELECT s.*
FROM AD_STUDENTS s
JOIN AD_EXAM_RESULTS er ON s.ID = er.STUDENT_ID
WHERE er.EXAM_GRADE > 95;

-- 3. Display the details of the students who have scored between 65 and 70
SELECT s.*
FROM AD_STUDENTS s
JOIN AD_EXAM_RESULTS er ON s.ID = er.STUDENT_ID
WHERE er.EXAM_GRADE BETWEEN 65 AND 70;

-- 4. Display the students who registered after 01-Jun-2012
SELECT *
FROM AD_STUDENTS
WHERE REG_YEAR > TO_DATE('01-JUN-2012', 'DD-MON-YYYY');
```

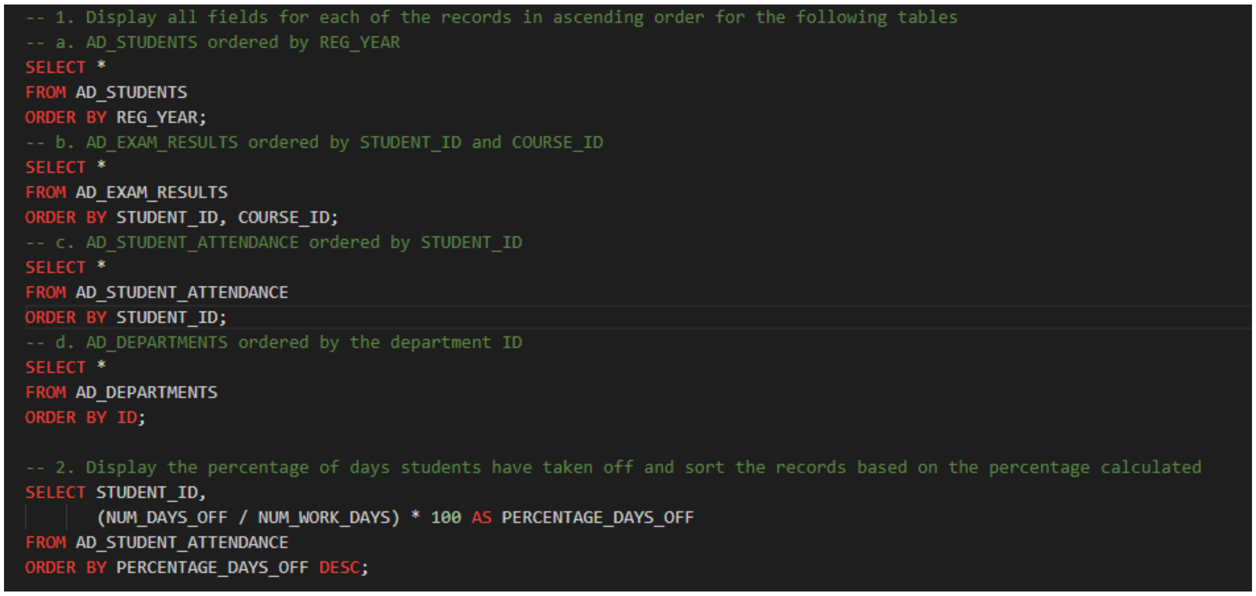


```sql
-- 5. Display the course details for departments 10 and 30
SELECT *
FROM AD_COURSES
WHERE DEPT_ID IN (10, 30);

-- 6. Display the details of students whose first name begins with the letter "J"
SELECT *
FROM AD_STUDENTS
WHERE FIRST_NAME LIKE 'J%';

-- 7. Display the details of students who have opted for courses 190 or 193
SELECT s.*
FROM AD_STUDENTS s
JOIN AD_STUDENT_COURSE_DETAILS scd ON s.ID = scd.STUDENT_ID
WHERE scd.COURSE_ID IN (190, 193);

-- 8. 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;
```

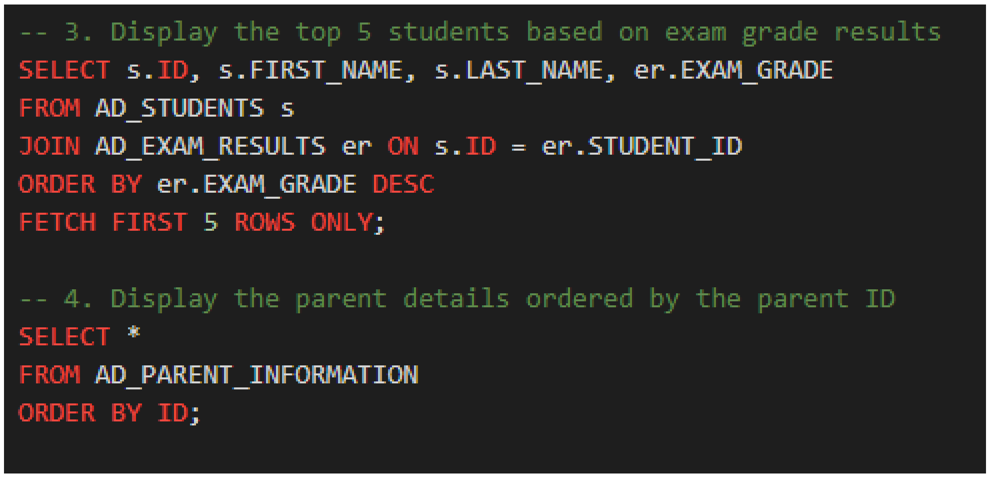


```sql
-- 9. Display the course details of courses not being offered in the summer and fall session (Session ID 200 and 300)
SELECT *
FROM AD_COURSES
WHERE SESSION_ID NOT IN (200, 300);

-- 10. Display the course details for department 20
SELECT *
FROM AD_COURSES
WHERE DEPT_ID = 20;
```

6-8 : Sorting Data Using ORDER BY Practices

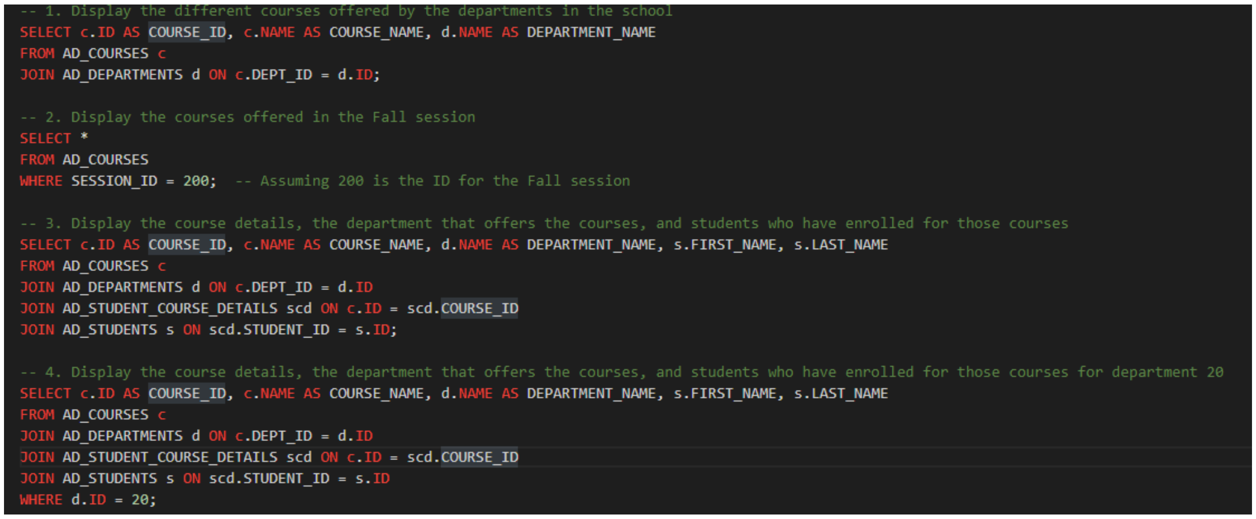


```sql
-- 1. Display all fields for each of the records in ascending order for the following tables
-- a. AD_STUDENTS ordered by REG_YEAR
SELECT *
FROM AD_STUDENTS
ORDER BY REG_YEAR;
-- b. AD_EXAM_RESULTS ordered by STUDENT_ID and COURSE_ID
SELECT *
FROM AD_EXAM_RESULTS
ORDER BY STUDENT_ID, COURSE_ID;
-- 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 ID;

-- 2. Display the percentage of days students have taken off and sort the records based on the percentage calculated
SELECT STUDENT_ID,
        (NUM_DAYS_OFF / NUM_WORK_DAYS) * 100 AS PERCENTAGE_DAYS_OFF
FROM AD_STUDENT_ATTENDANCE
ORDER BY PERCENTAGE_DAYS_OFF DESC;
```

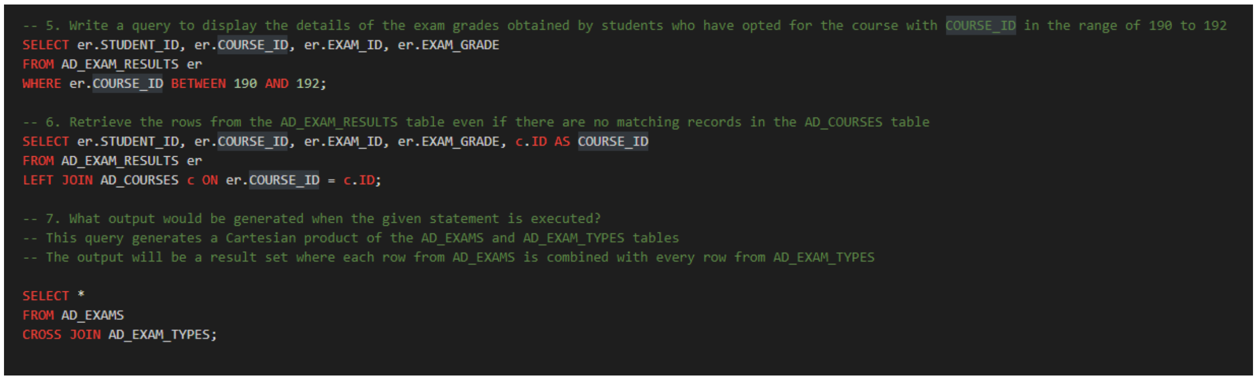


```sql
-- 3. Display the top 5 students based on exam grade results
SELECT s.ID, s.FIRST_NAME, s.LAST_NAME, er.EXAM_GRADE
FROM AD_STUDENTS s
JOIN AD_EXAM_RESULTS er ON s.ID = er.STUDENT_ID
ORDER BY er.EXAM_GRADE DESC
FETCH FIRST 5 ROWS ONLY;

-- 4. Display the parent details ordered by the parent ID
SELECT *
FROM AD_PARENT_INFORMATION
ORDER BY ID;
```

6-9 : Joining Tables Using JOIN Practices



```sql
-- 1. Display the different courses offered by the departments in the school
SELECT c.ID AS COURSE_ID, c.NAME AS COURSE_NAME, d.NAME AS DEPARTMENT_NAME
FROM AD_COURSES c
JOIN AD_DEPARTMENTS d ON c.DEPT_ID = d.ID;

-- 2. Display the courses offered in the Fall session
SELECT *
FROM AD_COURSES
WHERE SESSION_ID = 200;  -- Assuming 200 is the ID for the Fall session

-- 3. Display the course details, the department that offers the courses, and students who have enrolled for those courses
SELECT c.ID AS COURSE_ID, c.NAME AS COURSE_NAME, d.NAME AS DEPARTMENT_NAME, s.FIRST_NAME, s.LAST_NAME
FROM AD_COURSES c
JOIN AD_DEPARTMENTS d ON c.DEPT_ID = d.ID
JOIN AD_STUDENT_COURSE_DETAILS scd ON c.ID = scd.COURSE_ID
JOIN AD_STUDENTS s ON scd.STUDENT_ID = s.ID;

-- 4. Display the course details, the department that offers the courses, and students who have enrolled for those courses for department 20
SELECT c.ID AS COURSE_ID, c.NAME AS COURSE_NAME, d.NAME AS DEPARTMENT_NAME, s.FIRST_NAME, s.LAST_NAME
FROM AD_COURSES c
JOIN AD_DEPARTMENTS d ON c.DEPT_ID = d.ID
JOIN AD_STUDENT_COURSE_DETAILS scd ON c.ID = scd.COURSE_ID
JOIN AD_STUDENTS s ON scd.STUDENT_ID = s.ID
WHERE d.ID = 20;
```



```sql
-- 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 er.STUDENT_ID, er.COURSE_ID, er.EXAM_ID, er.EXAM_GRADE
FROM AD_EXAM_RESULTS er
WHERE er.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 er.STUDENT_ID, er.COURSE_ID, er.EXAM_ID, er.EXAM_GRADE, c.ID AS COURSE_ID
FROM AD_EXAM_RESULTS er
LEFT JOIN AD_COURSES c ON er.COURSE_ID = c.ID;

-- 7. What output would be generated when the given statement is executed?
-- This query generates a Cartesian product of the AD_EXAMS and AD_EXAM_TYPES tables
-- The output will be a result set where each row from AD_EXAMS is combined with every row from AD_EXAM_TYPES

SELECT *
FROM AD_EXAMS
CROSS JOIN AD_EXAM_TYPES;
```