5-1: Mapping Entities and Attributes Practice

This section emphasizes creating glossaries and applying naming standards in database design.

1. **Exercise 1: Creating a Glossary from the Logical Model**
    - Create a glossary from the academic database logical model.
    - Define the glossary's name, description, and classification types.
2. **Exercise 2: Forward Engineering the Design to Apply the Glossary and Naming Standard**
    - Add the glossary to the naming standards page.
    - Apply naming standards during the forward engineering process to the relational model.

5-2: Mapping Primary and Foreign Keys Practice

This section is centered on mapping unique identifiers and defining naming templates for primary and foreign keys.

1. **Exercise 1: Observe Mapping of Unique Identifiers and Relationships**
    - Compare the logical and relational models to verify the mapping of primary, unique, and foreign keys.
2. **Exercise 2: Define Table Name Abbreviations in CSV File**
    - Create a CSV file with abbreviations for table names.
3. **Exercise 3: Define Name Template**
    - Set templates for naming keys, constraints, and indexes using predefined variables.
4. **Exercise 4: Apply Name Template to Relational Model**
    - Apply the naming template to the entire relational model using the defined abbreviations.
5. **Exercise 5: Select Subtype Generation Method in Relational Model**
    - Define how subtypes are mapped to the relational model by selecting single table generation.

6-1: Introduction to Oracle Application Express

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

    - Registered for iAcademy APEX Learner Guide.

2. **Exercise 1:** Follow the Guide to learn about the features of Oracle Application Express.

- Thoroughly went through the guide to explore the features and possibilities available in Oracle Application Express.
- Explored different components such as App Builder & SQL Workshop and learnt about different terminologies related to APEX.

6-2: Structured Query Language

SQL short for Structured Query Language is the set-based, declarative language used to access data in a relational database. SQL is efficient and powerful giving the user ability to perform tasks to manipulate data. It keeps the databases consistent.

1. **Exercise 1:** Access and login to Oracle Application Express

    - Registered and gained access to Oracle Application Express.

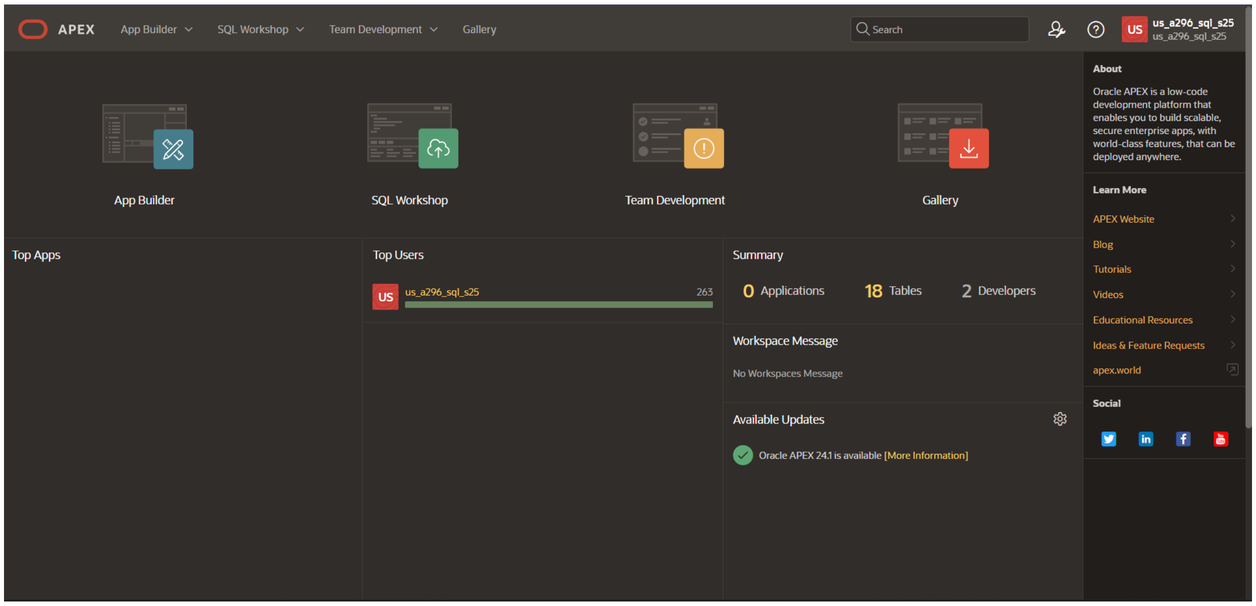
2. **Exercise 2:** Click the **Help** icon, and become familiar with the following section and topics:

    i) Application Express SQL Workshop:
    - SQL Workshop is a powerful feature within Oracle APEX that allows you to interact with the database, run SQL commands, and manage database objects. Below are the subsections you need to explore:

        a) Managing Database Objects with Object Browser:
        - The Object Browser is a tool that provides an interface for viewing, creating and managing database objects such as tables, views, and indexes.
        - Key features include Navigation, Viewing Details & Editing Objects.

        b) Using SQL Commands:
        - The APEX environment allows you to directly run SQL queries and updates.
        - Key features include SQL Commands and Output Viewer. You can also save commands for future use

        c) Using SQL Scripts:
        - Using SQL Scripts, you can create, save and execute larger scripts that include multiple commands.
        - Script management allows you to handle complex and big databases with relative ease.



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 Apps

Top Users

US us_a296_sql_s25 263

Summary

0 Applications 18 Tables 2 Developers

Workspace Message

No Workspaces Message

Available Updates

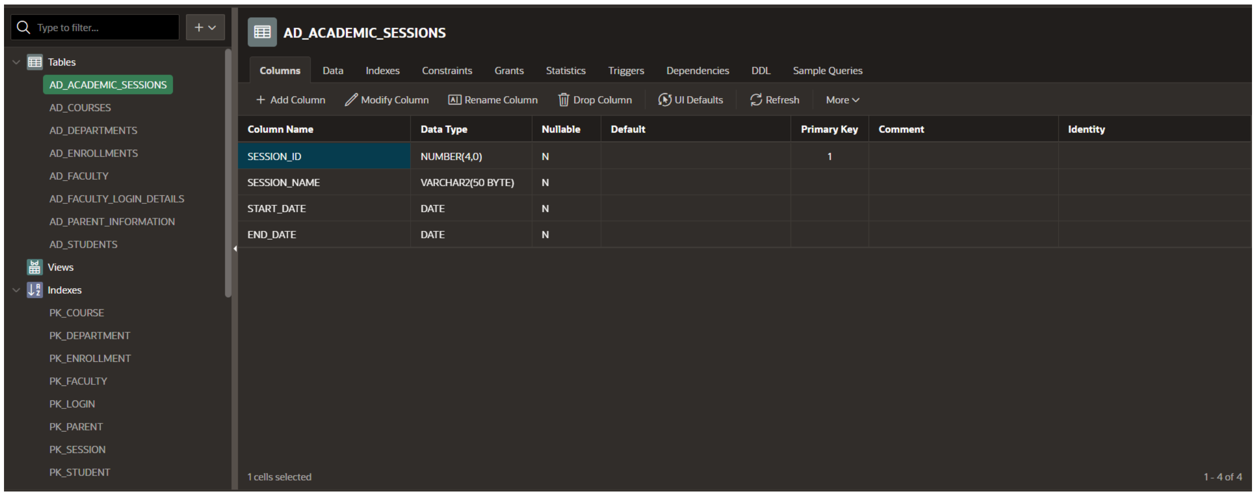
Oracle APEX 24.1 is available [More Information]

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

Data Definition Language (DDL) refers to the commands used to define and manage the structure of databases, such as creating, modifying and deleting objects in a database. The commands are specifically written in SQL.

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

    - Created 8 different tables using the DDL Statements for the **academic_db** with 13 different indexes.
    - Specified NOT NULL constraints wherever necessary.



AD_ACADEMIC_SESSIONS

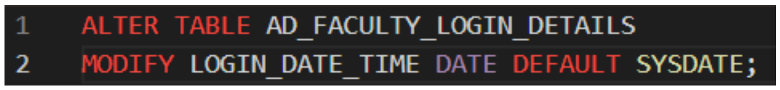
Column Name	Data Type	Nullable	Default	Primary Key	Comment	Identity
SESSION_ID	NUMBER(4,0)	N		1		
SESSION_NAME	VARCHAR2(50 BYTE)	N				
START_DATE	DATE	N				
END_DATE	DATE	N				

1 cells selected

1 - 4 of 4

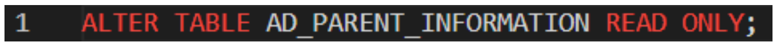
2. **Exercise 2:** Altering the tables,

1. Alter the tables in the Academic Database to define the primary key, foreign key and unique constraints.
    - Altered the tables for definition of unique constraints for each table in the academic_db.
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 DATE DEFAULT SYSDATE;
```

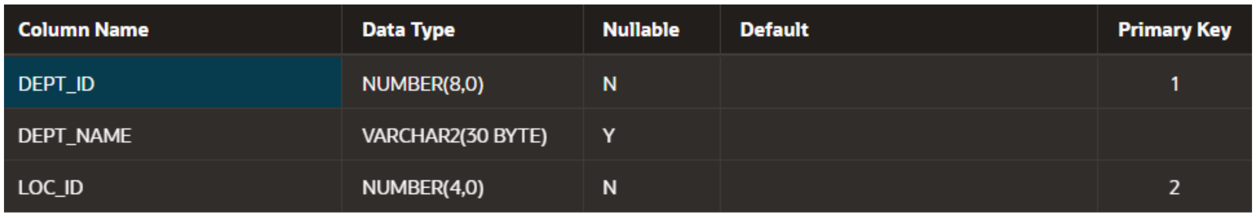
3. Set the AD_PARENT_INFORMATION table to a read-only status.



```
ALTER TABLE AD_PARENT_INFORMATION READ ONLY;
```

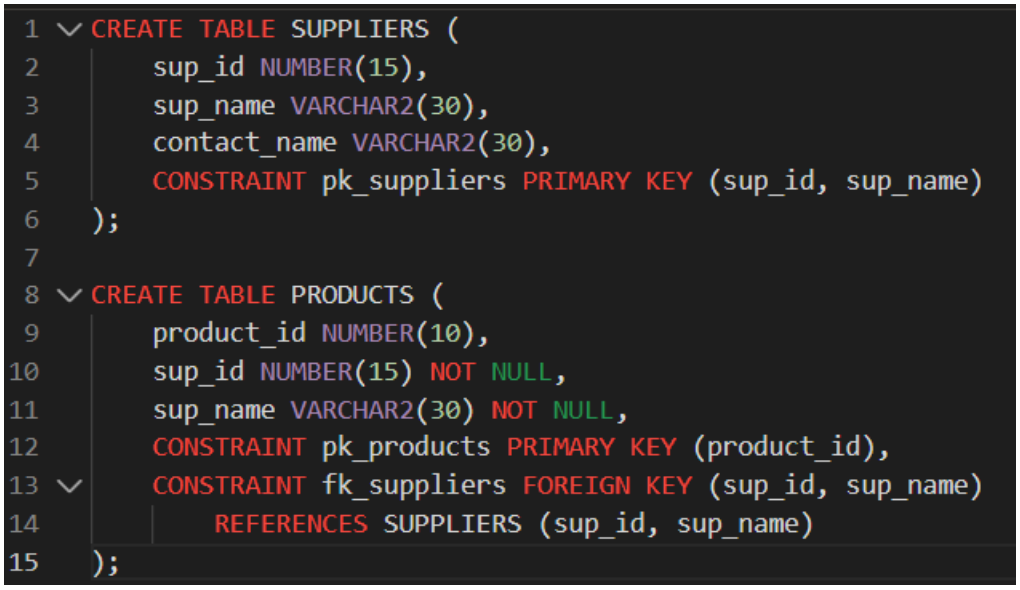
3. **Exercise 3:** Creating Composite, Primary & Foreign Keys.

1. Create Dept Table, the primary key for this table needs to be defined as composite comprising of the dept_id and loc_id.
    - Created a dept table as per requirements.



Column Name	Data Type	Nullable	Default	Primary Key
DEPT_ID	NUMBER(8,0)	N		1
DEPT_NAME	VARCHAR2(30 BYTE)	Y		
LOC_ID	NUMBER(4,0)	N		2

2. Create the SUPPLIERS and PRODUCTS table.
    - Created two new tables SUPPLIERS & PRODUCTS.



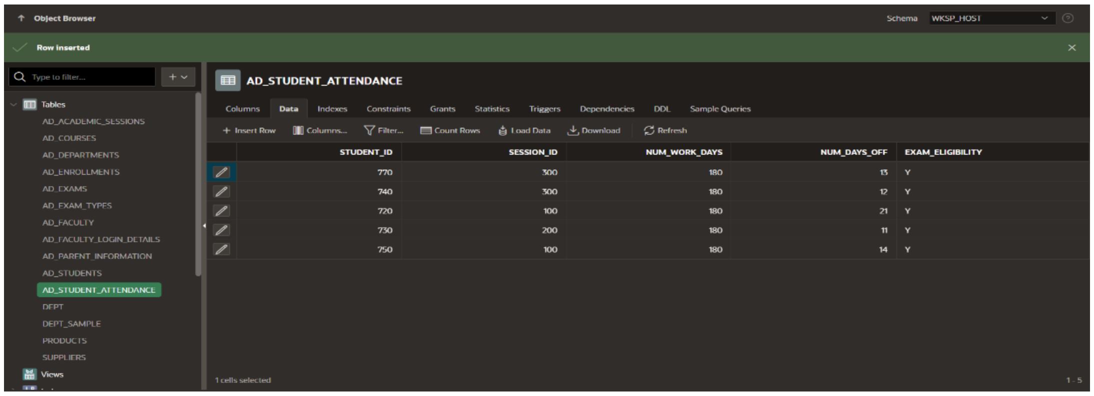
```sql
CREATE TABLE SUPPLIERS (
    sup_id NUMBER(15),
    sup_name VARCHAR2(30),
    contact_name VARCHAR2(30),
    CONSTRAINT pk_suppliers PRIMARY KEY (sup_id, sup_name)
);

CREATE TABLE PRODUCTS (
    product_id NUMBER(10),
    sup_id NUMBER(15) NOT NULL,
    sup_name VARCHAR2(30) NOT NULL,
    CONSTRAINT pk_products PRIMARY KEY (product_id),
    CONSTRAINT fk_suppliers FOREIGN KEY (sup_id, sup_name)
        REFERENCES SUPPLIERS (sup_id, sup_name)
);
```

6-4: Defining Data Manipulation

Data Definition Language (DDL) refers to the commands used to define and manage the structure of databases, such as creating, modifying and deleting objects in a database. The commands are specifically written in SQL.

1. **Exercise 1:** Inserting rows into tables.

    - Inserted data into rows in all the tables AD_ACADEMIC_SESSIONS, AD_DEPARTMENTS, AD_PARENT_INFORMATION, AD_STUDENTS, AD_COURSES, AD_FACULTY, AD_EXAM_TYPES, AD_EXAMS, AD_STUDENT_ATTENDANCE, AD_FACULTY_COURSE_DETAILS and AD_FACULTY_LOGIN_DETAILS



Row inserted

AD_STUDENT_ATTENDANCE

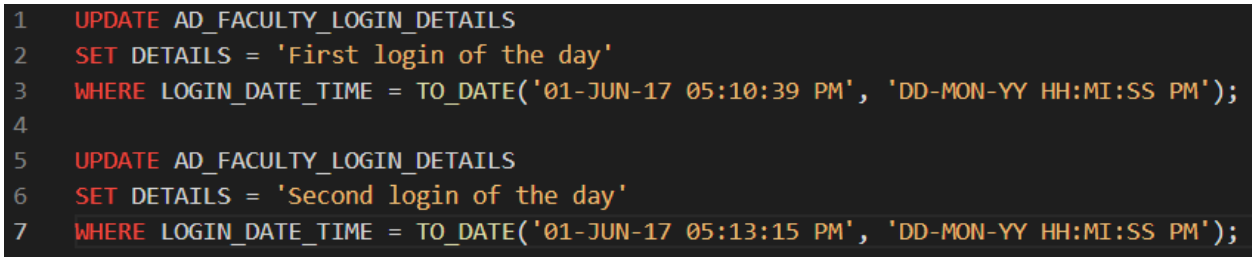
STUDENT_ID	SESSION_ID	NUM_WORK_DAYS	NUM_DAYS_OFF	EXAM_ELIGIBILITY
770	300	180	13	Y
740	300	180	12	Y
720	100	180	21	Y
730	200	180	11	Y
750	100	180	14	Y

1 cells selected

1 - 5

2. **Exercise 2:** Updating rows in the table

- Updated rows in the FACULTY_LOGIN_DETAILS table and made a field VARCHAR2(50) and possible to have null.



```sql
UPDATE AD_FACULTY_LOGIN_DETAILS
SET DETAILS = 'First login of the day'
WHERE LOGIN_DATE_TIME = TO_DATE('01-JUN-17 05:10:39 PM', 'DD-MON-YY HH:MI:SS PM');

UPDATE AD_FACULTY_LOGIN_DETAILS
SET DETAILS = 'Second login of the day'
WHERE LOGIN_DATE_TIME = TO_DATE('01-JUN-17 05:13:15 PM', 'DD-MON-YY HH:MI:SS PM');
```

6-5: Defining Transaction Control Practices

Transaction Control Practices are necessary to maintain data integrity and data consistency in a database. It allows you to implement SQL operations as a single unit of work, making sure that either all operations succeed or none do.

1. **Exercise 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); 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 Save point is created called ALTER_DONE. A ROLLBACK is issued after the Save point ALTER_DONE. Would the new email field still be there?

    - After issuing the rollback to the save point ALTER_DONE, the EMAIL_ADDR column will no longer exist in the AD_STUDENT_TEST_DETAILS table. The rollback operation reverts the table to its state before the EMAIL_ADDR column was added.

2. **Exercise 2:** Updating rows in the table

    1. If an INSERT is done to add rows into the test table and a Save point is then created called INSERT_DONE. Then an UPDATE to a row in the test table is done and a Save point is created called UPDATE_DONE. Then a DELETE is executed to delete a row in the test table and a Save point is created called DELETE_DONE. At this point what records would be in the table? Then a ROLLBACK to Save point UPDATE_DONE is issued. What changes would you notice with respect to the transactions and the records remaining in the table?
        The final State of the table would be:
        1. `(920, 'MAC', TO_DATE('01-Jan-2012', 'DD-MON-YYYY'), NULL)`
        2. `(930, 'MACC', TO_DATE('01-Jan-2012', 'DD-MON-YYYY'), NULL)`

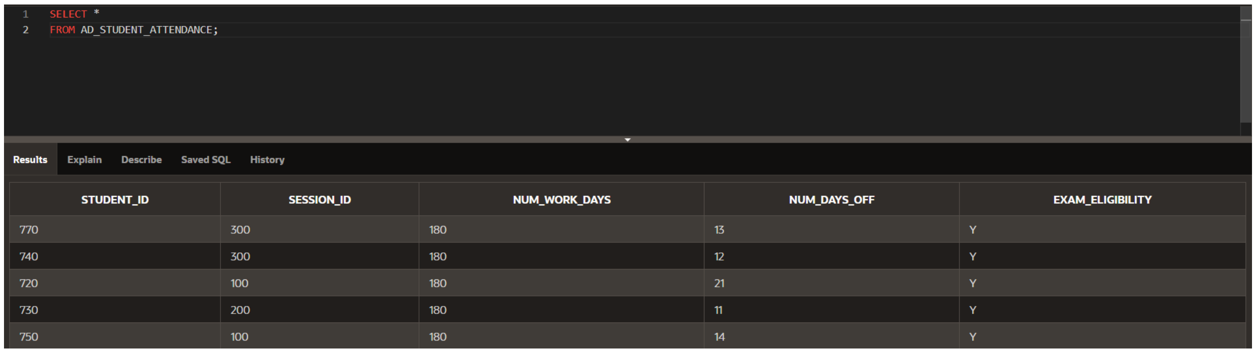
3. `(940, 'MACCC', TO_DATE('01-Jan-2012', 'DD-MON-YYYY'), 'Mac@abc.com')`
4. `(950, 'MACCC', TO_DATE('01-Jan-2012', 'DD-MON-YYYY'), NULL)`
5. `(920, 'MACCCC', TO_DATE('01-Jan-2012', 'DD-MON-YYYY'), NULL)`

6-6: Retrieving Data Practices

Retrieving data from a database is fundamental to working with relational databases. SQL is a really powerful tool which can perform any task.

1. **Exercise 1:** Retrieving columns from tables.

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



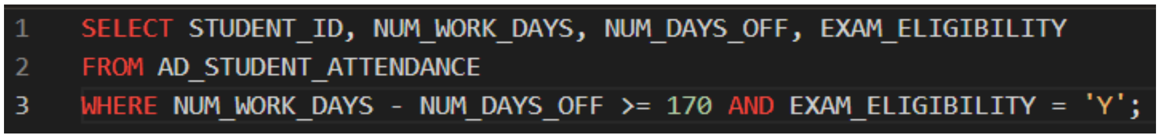
```
SELECT *
FROM AD_STUDENT_ATTENDANCE;
```

Results Explain Describe Saved SQL History

STUDENT_ID	SESSION_ID	NUM_WORK_DAYS	NUM_DAYS_OFF	EXAM_ELIGIBILITY
770	300	180	13	Y
740	300	180	12	Y
720	100	180	21	Y
730	200	180	11	Y
750	100	180	14	Y

- Write a query to retrieve the exam grade obtained by each student for every exam attempted.
    1. SELECT STUDENT_ID, EXAM_ID, EXAM_GRADE FROM AD_EXAM_RESULTS ORDER BY STUDENT_ID, EXAM_ID;

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

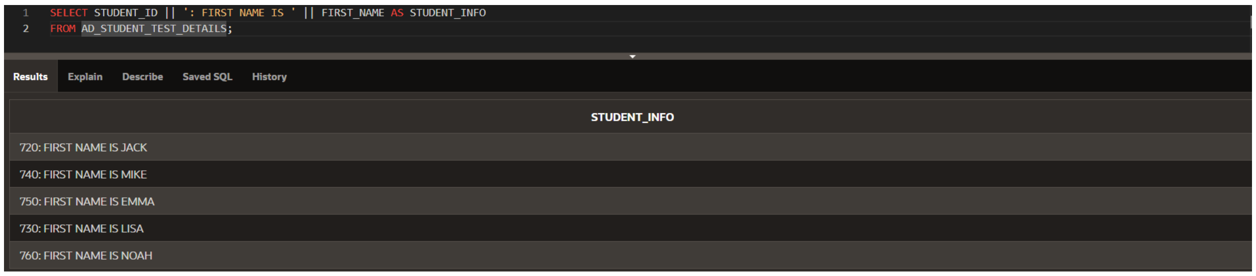


```
SELECT STUDENT_ID, NUM_WORK_DAYS, NUM_DAYS_OFF, EXAM_ELIGIBILITY
FROM AD_STUDENT_ATTENDANCE
WHERE NUM_WORK_DAYS - NUM_DAYS_OFF >= 170 AND EXAM_ELIGIBILITY = 'Y';
```

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

1. SELECT DEPARTMENT_NAME, HEAD AS HEAD_NAME FROM AD_DEPARTMENTS;

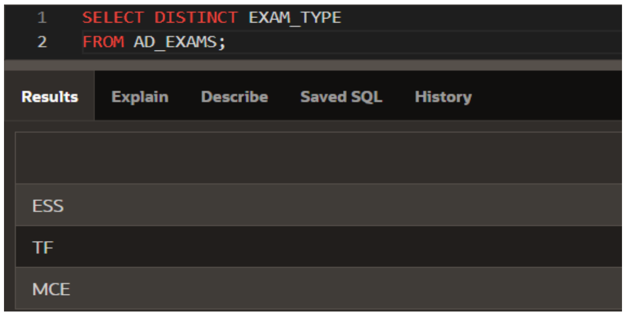
- Retrieve the student ID and first name for each student concatenated with literal text to look like this: 720: FIRST NAME IS JACK.



```
SELECT STUDENT_ID || ': FIRST NAME IS ' || FIRST_NAME AS STUDENT_INFO
FROM AD_STUDENT_TEST_DETAILS;
```

```
STUDENT_INFO
720: FIRST NAME IS JACK
740: FIRST NAME IS MIKE
750: FIRST NAME IS EMMA
730: FIRST NAME IS LISA
760: FIRST NAME IS NOAH
```

- Display all the distinct exam types from the AD_EXAMS table



```
SELECT DISTINCT EXAM_TYPE
FROM AD_EXAMS;
```

```
ESS
TF
MCE
```

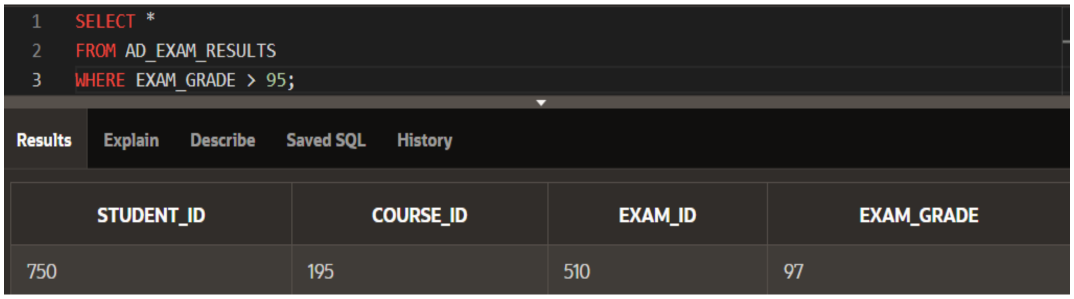
6-7: Restricting Data Using WHERE Statement Practices

Taking use of WHERE statement allows you to restrict the rows returned by your queries based on conditions.

1. **Exercise 1:** Restricting data using SELECT.

- Display the course details for the Spring Season.

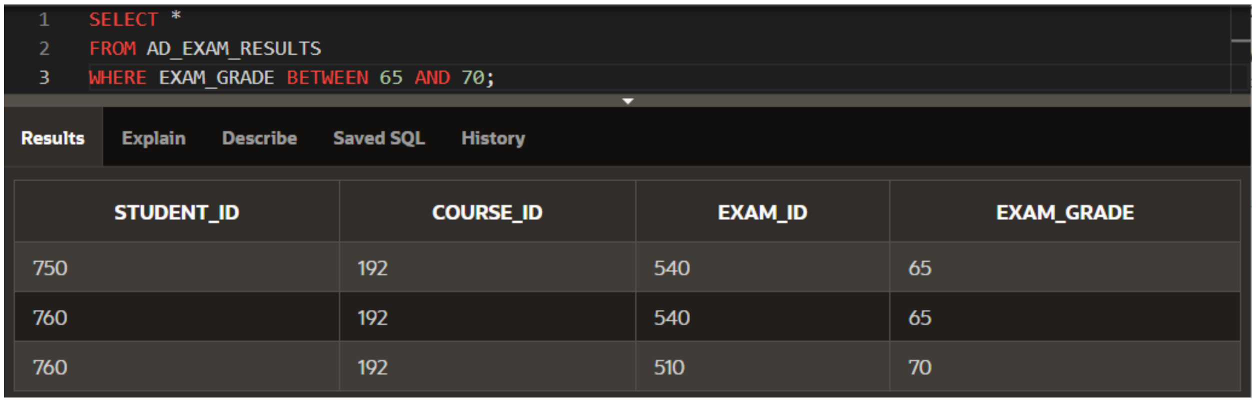
  SELECT c.* FROM AD_COURSES c JOIN AD_ACADEMIC_SESSIONS s ON c.SESSION_ID = s.SESSION_ID WHERE s.SESSION_NAME = 'SPRING SESSION';
- Display the details of the students who have scored more than 95.



```sql
SELECT *
FROM AD_EXAM_RESULTS
WHERE EXAM_GRADE > 95;
```

STUDENT_ID	COURSE_ID	EXAM_ID	EXAM_GRADE
750	195	510	97

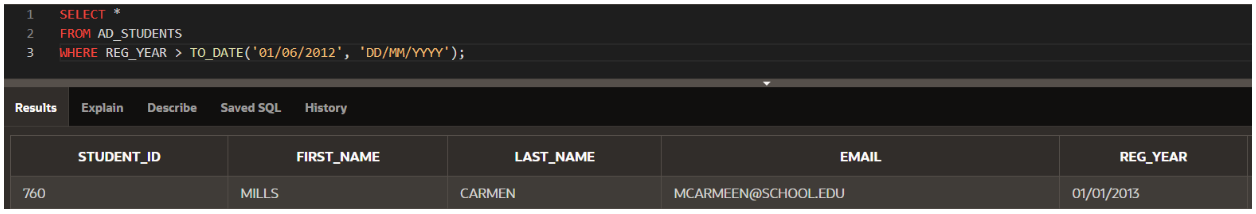
- Display the details of the students who have scored between 65 and 70.



```sql
SELECT *
FROM AD_EXAM_RESULTS
WHERE EXAM_GRADE BETWEEN 65 AND 70;
```

STUDENT_ID	COURSE_ID	EXAM_ID	EXAM_GRADE
750	192	540	65
760	192	540	65
760	192	510	70

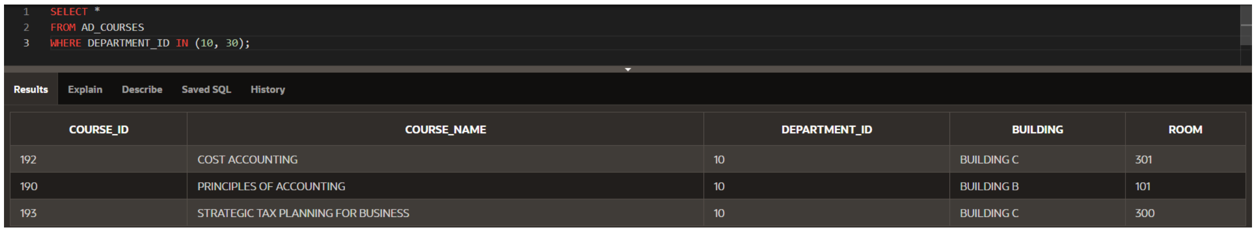
- Display the students who registered after 01-Jun-2012.



```sql
SELECT *
FROM AD_STUDENTS
WHERE REG_YEAR > TO_DATE('01/06/2012', 'DD/MM/YYYY');
```

STUDENT_ID	FIRST_NAME	LAST_NAME	EMAIL	REG_YEAR
760	MILLS	CARMEN	MCARMEEN@SCHOOL.EDU	01/01/2013

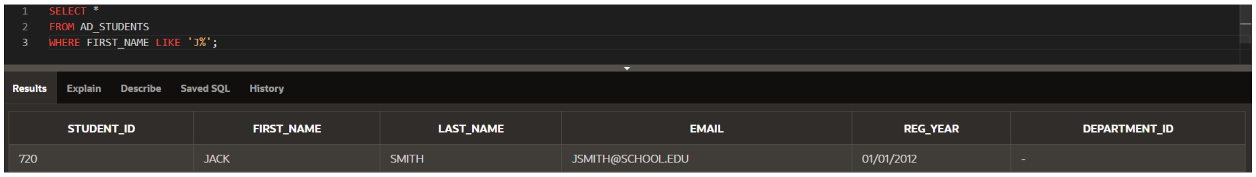
- Display the course details for departments 10 and 30.



```sql
SELECT *
FROM AD_COURSES
WHERE DEPARTMENT_ID IN (10, 30);
```

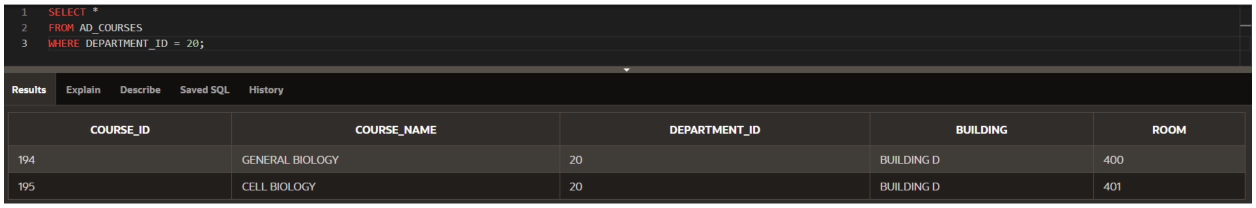
COURSE_ID	COURSE_NAME	DEPARTMENT_ID	BUILDING	ROOM
192	COST ACCOUNTING	10	BUILDING C	301
190	PRINCIPLES OF ACCOUNTING	10	BUILDING B	101
193	STRATEGIC TAX PLANNING FOR BUSINESS	10	BUILDING C	300

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


```
SELECT *
FROM AD_STUDENTS
WHERE FIRST_NAME LIKE 'J%';
```

STUDENT_ID	FIRST_NAME	LAST_NAME	EMAIL	REG_YEAR	DEPARTMENT_ID
720	JACK	SMITH	JSMITH@SCHOOL.EDU	01/01/2012	-

- Display the course details for department 20.


```
SELECT *
FROM AD_COURSES
WHERE DEPARTMENT_ID = 20;
```

COURSE_ID	COURSE_NAME	DEPARTMENT_ID	BUILDING	ROOM
194	GENERAL BIOLOGY	20	BUILDING D	400
195	CELL BIOLOGY	20	BUILDING D	401

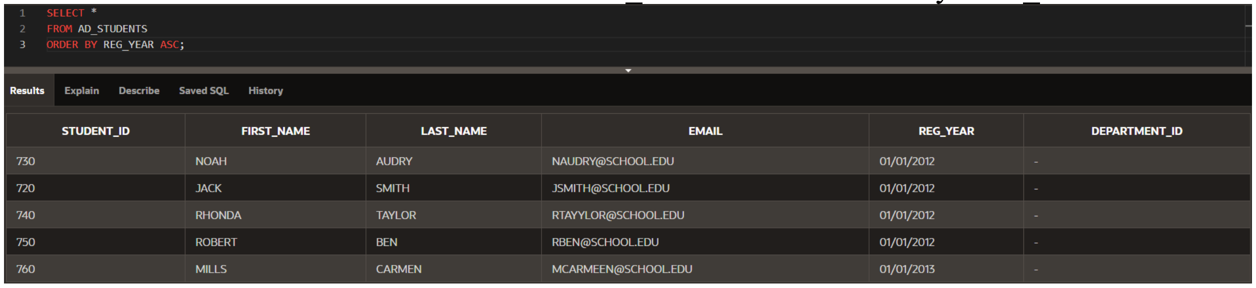
6-8: Sorting data by using ORDER BY

The ORDER BY clause in SQL is used to sort the results of a query

1. **Exercise 1:** Sorting data using ORDER BY.

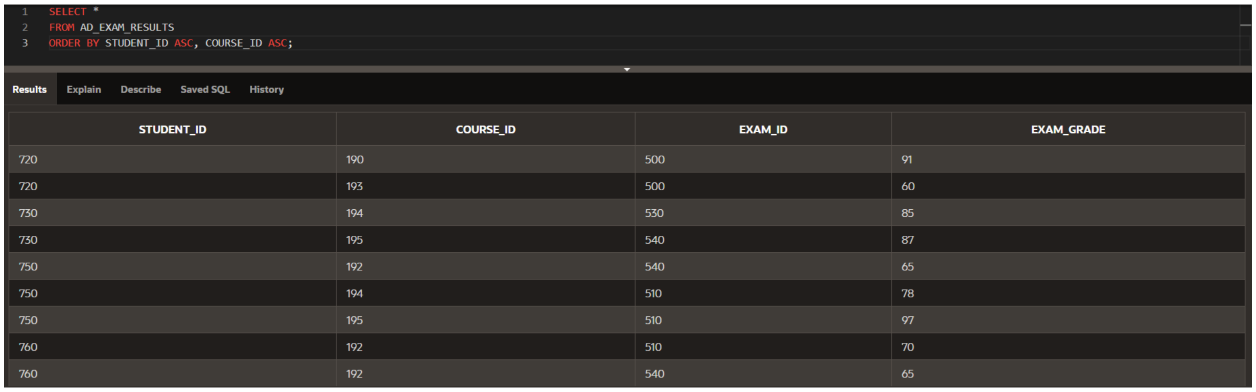
   - Display all fields for each of the records in ascending order for the following tables:

     a. AD_STUDENTS ordered by REG_YEAR


```
SELECT *
FROM AD_STUDENTS
ORDER BY REG_YEAR ASC;
```

STUDENT_ID	FIRST_NAME	LAST_NAME	EMAIL	REG_YEAR	DEPARTMENT_ID
730	NOAH	AUDRY	NAUDRY@SCHOOL.EDU	01/01/2012	-
720	JACK	SMITH	JSMITH@SCHOOL.EDU	01/01/2012	-
740	RHONDA	TAYLOR	RTAYYLOR@SCHOOL.EDU	01/01/2012	-
750	ROBERT	BEN	RBEN@SCHOOL.EDU	01/01/2012	-
760	MILLS	CARMEN	MCARMEEN@SCHOOL.EDU	01/01/2013	-

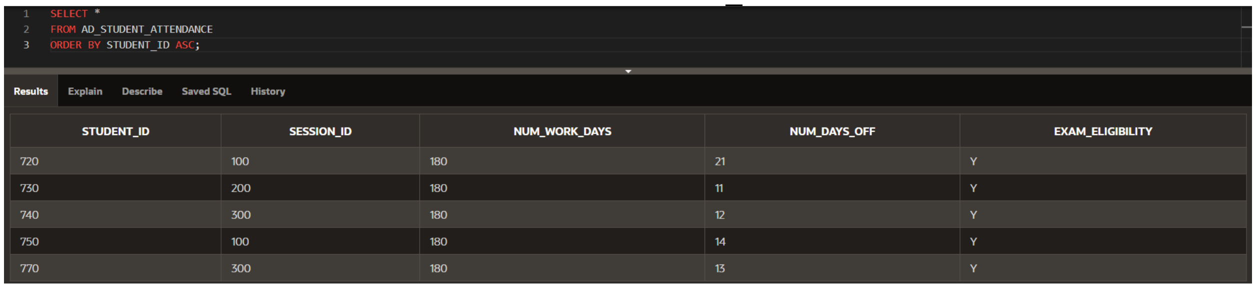
     b. AD_EXAM_RESULTS ordered by STUDENT_ID and COURSE_ID



```sql
SELECT *
FROM AD_EXAM_RESULTS
ORDER BY STUDENT_ID ASC, COURSE_ID ASC;
```

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

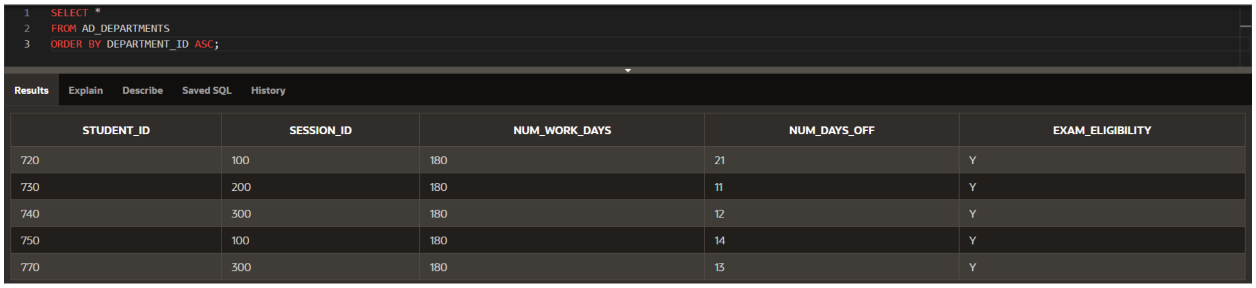
c. c. AD_STUDENT_ATTENDANCE ordered by STUDENT_ID



```sql
SELECT *
FROM AD_STUDENT_ATTENDANCE
ORDER BY STUDENT_ID ASC;
```

STUDENT_ID	SESSION_ID	NUM_WORK_DAYS	NUM_DAYS_OFF	EXAM_ELIGIBILITY
720	100	180	21	Y
730	200	180	11	Y
740	300	180	12	Y
750	100	180	14	Y
770	300	180	13	Y

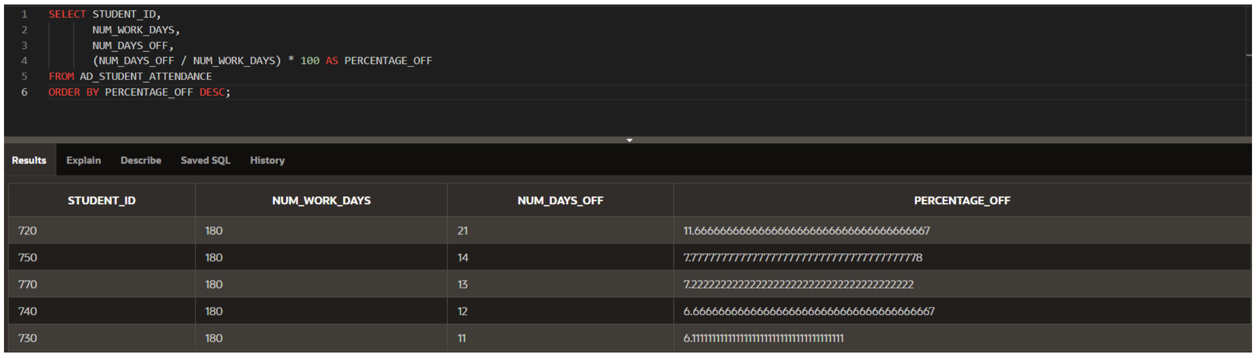
d. AD_DEPARTMENTS ordered by the department ID



```sql
SELECT *
FROM AD_DEPARTMENTS
ORDER BY DEPARTMENT_ID ASC;
```

STUDENT_ID	SESSION_ID	NUM_WORK_DAYS	NUM_DAYS_OFF	EXAM_ELIGIBILITY
720	100	180	21	Y
730	200	180	11	Y
740	300	180	12	Y
750	100	180	14	Y
770	300	180	13	Y

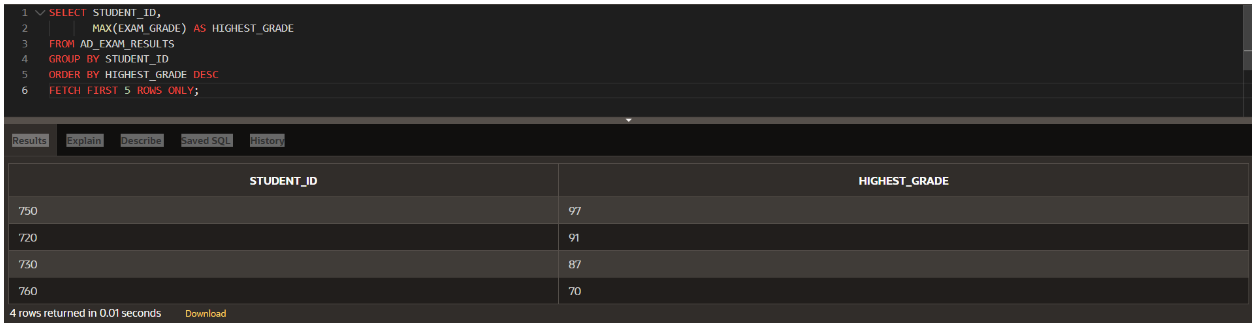
- Display the percentage of days' students have taken days off and sort the records based on the percentage calculated.



```sql
SELECT STUDENT_ID,
       NUM_WORK_DAYS,
       NUM_DAYS_OFF,
       (NUM_DAYS_OFF / NUM_WORK_DAYS) * 100 AS PERCENTAGE_OFF
FROM AD_STUDENT_ATTENDANCE
ORDER BY PERCENTAGE_OFF DESC;
```

STUDENT_ID	NUM_WORK_DAYS	NUM_DAYS_OFF	PERCENTAGE_OFF
720	180	21	11.66666666666666666666666666666666666667
750	180	14	7.777777777777777777777777777777777777778
770	180	13	7.222222222222222222222222222222222222222
740	180	12	6.666666666666666666666666666666666666667
730	180	11	6.111111111111111111111111111111111111111

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

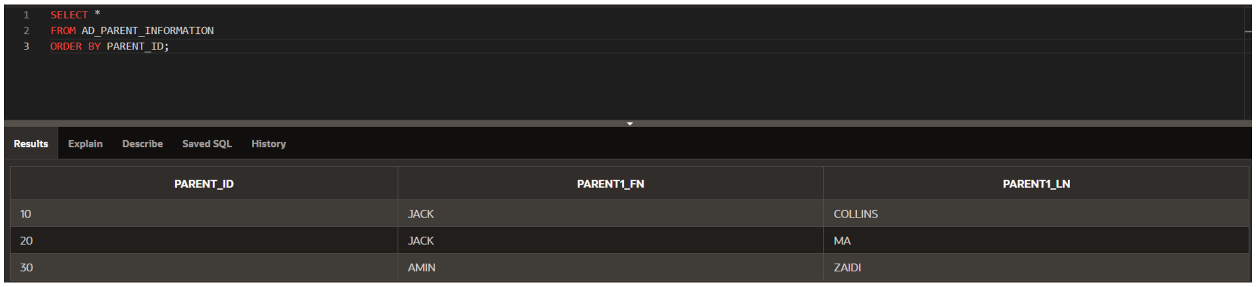


```
SELECT STUDENT_ID,
       MAX(EXAM_GRADE) AS HIGHEST_GRADE
FROM AD_EXAM_RESULTS
GROUP BY STUDENT_ID
ORDER BY HIGHEST_GRADE DESC
FETCH FIRST 5 ROWS ONLY;
```

STUDENT_ID	HIGHEST_GRADE
750	97
720	91
730	87
760	70

4 rows returned in 0.01 seconds   Download

- Display the parent details ordered by the parent ID



```
SELECT *
FROM AD_PARENT_INFORMATION
ORDER BY PARENT_ID;
```

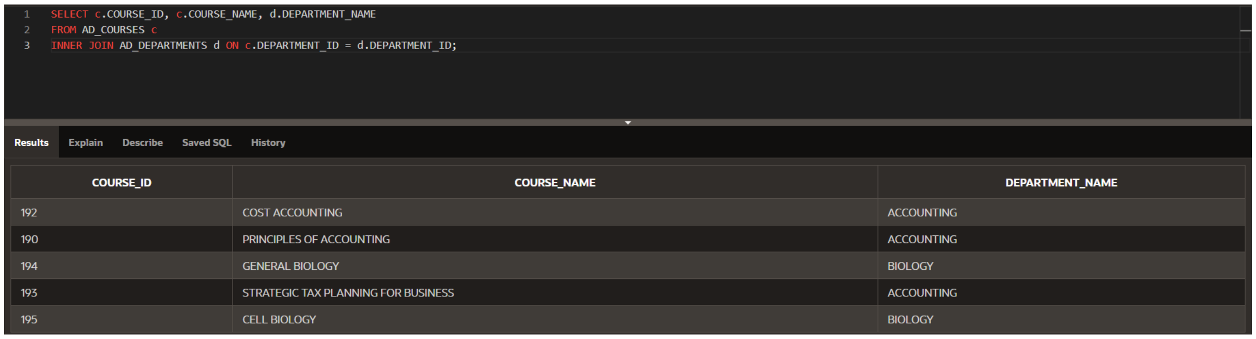
PARENT_ID	PARENT1_FN	PARENT1_LN
10	JACK	COLLINS
20	JACK	MA
30	AMIN	ZAIDI

6-9: Joining tables using JOINS in SQL

There are multiple JOINS in SQL which are used for joining tables with each other

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

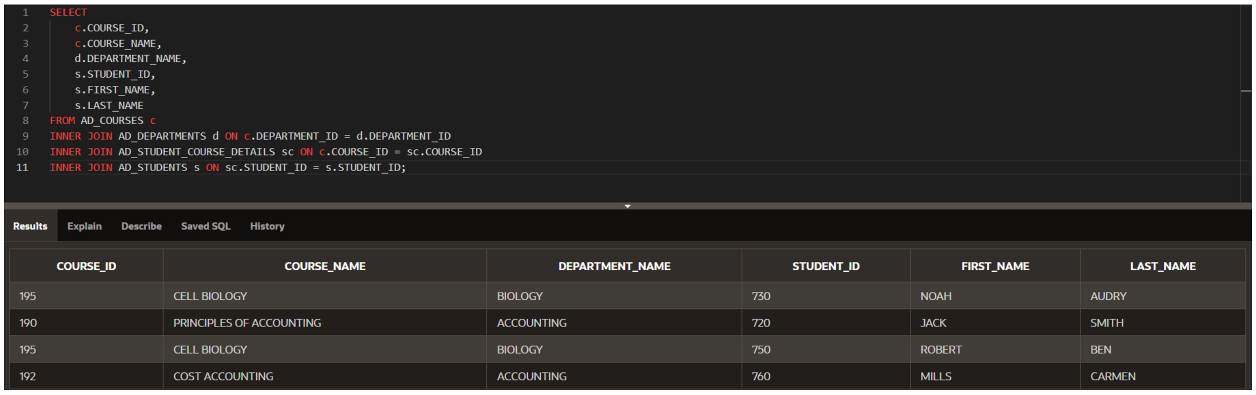
- Display the different courses offered by the departments in the school.



```
SELECT c.COURSE_ID, c.COURSE_NAME, d.DEPARTMENT_NAME
FROM AD_COURSES c
INNER JOIN AD_DEPARTMENTS d ON c.DEPARTMENT_ID = d.DEPARTMENT_ID;
```

COURSE_ID	COURSE_NAME	DEPARTMENT_NAME
192	COST ACCOUNTING	ACCOUNTING
190	PRINCIPLES OF ACCOUNTING	ACCOUNTING
194	GENERAL BIOLOGY	BIOLOGY
193	STRATEGIC TAX PLANNING FOR BUSINESS	ACCOUNTING
195	CELL BIOLOGY	BIOLOGY

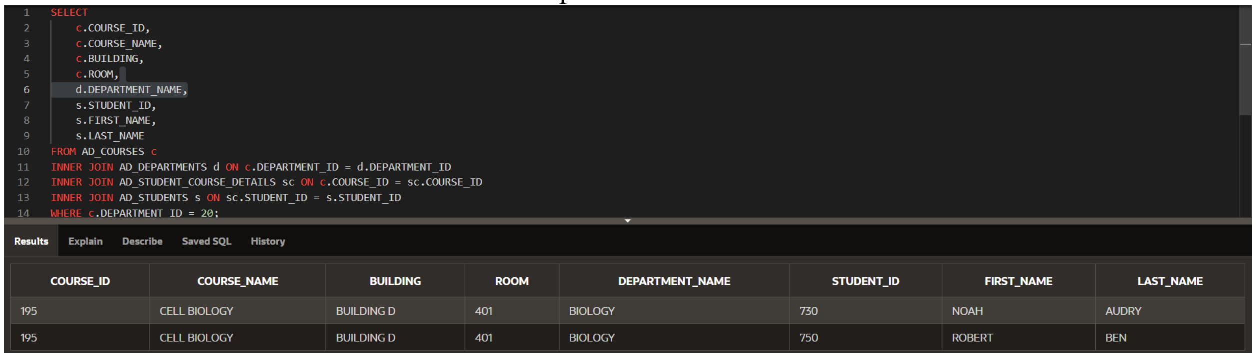
- Display the course details, the department that offers the courses and students who have enrolled for those courses.



```sql
SELECT
    c.COURSE_ID,
    c.COURSE_NAME,
    d.DEPARTMENT_NAME,
    s.STUDENT_ID,
    s.FIRST_NAME,
    s.LAST_NAME
FROM AD_COURSES c
INNER JOIN AD_DEPARTMENTS d ON c.DEPARTMENT_ID = d.DEPARTMENT_ID
INNER JOIN AD_STUDENT_COURSE_DETAILS sc ON c.COURSE_ID = sc.COURSE_ID
INNER JOIN AD_STUDENTS s ON sc.STUDENT_ID = s.STUDENT_ID;
```

COURSE_ID	COURSE_NAME	DEPARTMENT_NAME	STUDENT_ID	FIRST_NAME	LAST_NAME
195	CELL BIOLOGY	BIOLOGY	730	NOAH	AUDRY
190	PRINCIPLES OF ACCOUNTING	ACCOUNTING	720	JACK	SMITH
195	CELL BIOLOGY	BIOLOGY	750	ROBERT	BEN
192	COST ACCOUNTING	ACCOUNTING	760	MILLS	CARMEN

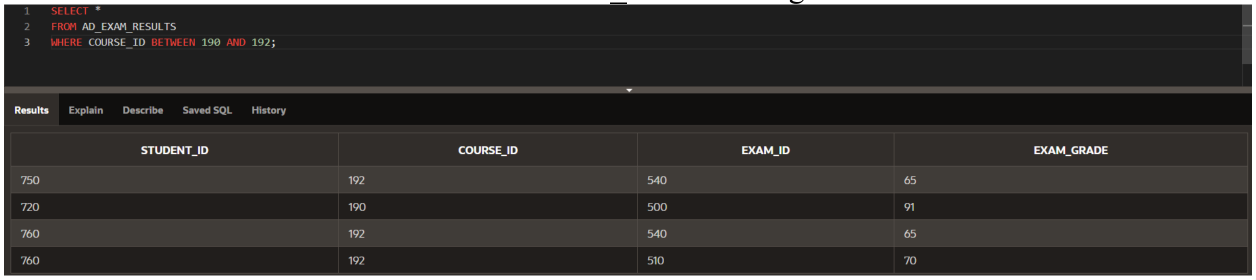
- Display the course details, the department that offers the courses and students who have enrolled for those courses for department 20.



```sql
SELECT
    c.COURSE_ID,
    c.COURSE_NAME,
    c.BUILDING,
    c.ROOM,
    d.DEPARTMENT_NAME,
    s.STUDENT_ID,
    s.FIRST_NAME,
    s.LAST_NAME
FROM AD_COURSES c
INNER JOIN AD_DEPARTMENTS d ON c.DEPARTMENT_ID = d.DEPARTMENT_ID
INNER JOIN AD_STUDENT_COURSE_DETAILS sc ON c.COURSE_ID = sc.COURSE_ID
INNER JOIN AD_STUDENTS s ON sc.STUDENT_ID = s.STUDENT_ID
WHERE c.DEPARTMENT_ID = 20;
```

COURSE_ID	COURSE_NAME	BUILDING	ROOM	DEPARTMENT_NAME	STUDENT_ID	FIRST_NAME	LAST_NAME
195	CELL BIOLOGY	BUILDING D	401	BIOLOGY	730	NOAH	AUDRY
195	CELL BIOLOGY	BUILDING D	401	BIOLOGY	750	ROBERT	BEN

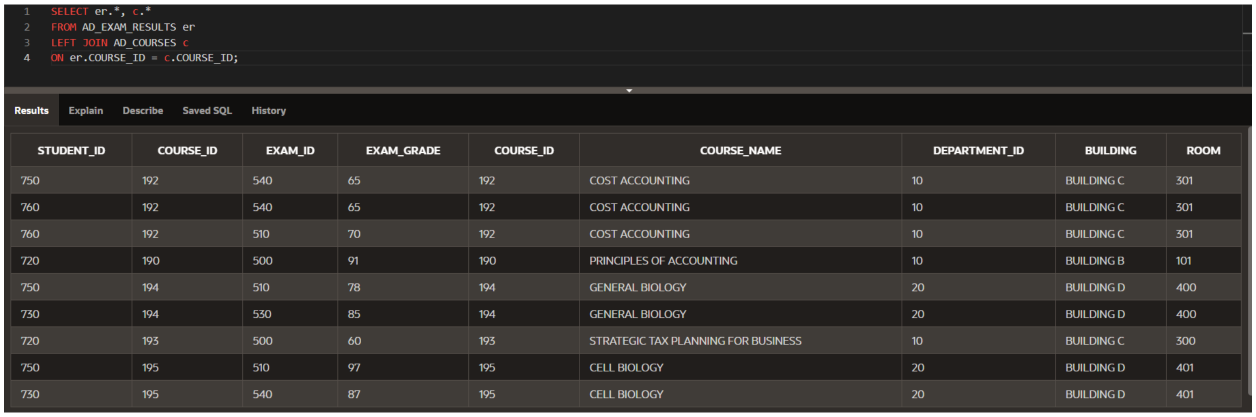
- Write a query to display the details of the exam grades obtained by students who have opted for the course with COURSE_ID in the range of 190 to 192.



```sql
SELECT *
FROM AD_EXAM_RESULTS
WHERE COURSE_ID BETWEEN 190 AND 192;
```

STUDENT_ID	COURSE_ID	EXAM_ID	EXAM_GRADE
750	192	540	65
720	190	500	91
760	192	540	65
760	192	510	70

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



```
SELECT er.*, c.*
FROM AD_EXAM_RESULTS er
LEFT JOIN AD_COURSES c
ON er.COURSE_ID = c.COURSE_ID;
```

STUDENT_ID	COURSE_ID	EXAM_ID	EXAM_GRADE	COURSE_ID	COURSE_NAME	DEPARTMENT_ID	BUILDING	ROOM
750	192	540	65	192	COST ACCOUNTING	10	BUILDING C	301
760	192	540	65	192	COST ACCOUNTING	10	BUILDING C	301
760	192	510	70	192	COST ACCOUNTING	10	BUILDING C	301
720	190	500	91	190	PRINCIPLES OF ACCOUNTING	10	BUILDING B	101
750	194	510	78	194	GENERAL BIOLOGY	20	BUILDING D	400
730	194	530	85	194	GENERAL BIOLOGY	20	BUILDING D	400
720	193	500	60	193	STRATEGIC TAX PLANNING FOR BUSINESS	10	BUILDING C	300
750	195	510	97	195	CELL BIOLOGY	20	BUILDING D	401
730	195	540	87	195	CELL BIOLOGY	20	BUILDING D	401

- What output would be generated when the given statement is executed? SELECT * FROM AD_EXAMS CROSS JOIN AD_EXAM_TYPES;
    a) Every row from AD_EXAMS table will be combined with every row from the AD_EXAM_TYPES table.