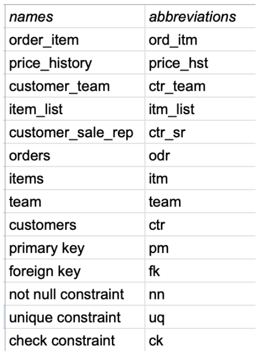
Vanessa Chen
HHA 507 - Oracle SQL Training
Dr. Hants Williams

Database Foundations 5-1: Mapping Entities and Attributes Practices

- *Exercise 1:* Creating a Glossary from the Logical Model
    - To complete the task, I used the provided code files to open the academic database and imported them. Next, I merged the files into the relational database and received the tree diagram. By filling out the required fields in the database, I was able to construct the glossary by clicking on the logical model button. It brought me to various options and I chose the create glossary file. The next step was using the tree diagram for the attributes and entities. Then, I noted the important and relevant abbreviations for the glossary.
- *Exercise 2:* Forward engineering the design to apply the Glossary and Naming Standard
    - To complete the task, I searched for logical model and found the properties option. I went to naming standards and searched for the glossary. I pressed the arrows to engineer the relational database and proceeded to general options. Then, I chose the option to apply name translation checkbox and made sure that use preferred abbreviations was the selected option. Following these steps allowed me to start engineering the model using the preferred abbreviations created and included in the glossary.

Database Foundations 5-2: Mapping Primary and Foreign Keys Practice

- *Exercise 1:* Observe the mapping of the unique identifiers and relationship in the Relational Model
    - From the tree diagram, there were sections underneath each table that specified the constraints regarding the primary key and foreign key relationships. The goal is to create a spreadsheet or csv file that will detail the relationship of the primary keys and unique keys to the foreign keys. For each entity, I created a table that consisted of the plural form of the attribute names and relevant abbreviations.
- *Exercise 2:* Define table name abbreviations in csv file
    - The image attached below is the csv file containing the relevant abbreviations and plural form of the attribute names for the physical data model layer



names	*abbreviations*
order_item	ord_itm
price_history	price_hst
customer_team	ctr_team
item_list	itm_list
customer_sale_rep	ctr_sr
orders	odr
items	itm
team	team
customers	ctr
primary key	pm
foreign key	fk
not null constraint	nn
unique constraint	uq
check constraint	ck

- *Exercise 3:* Define Name Template
  - To complete the task of setting a template for keys and constraints, I took the developed csv file and combined it with the predefined variables by choosing academic database design in the object browser. I proceeded with the properties option and chose settings. From there, naming standard was selected and then the templates option was chosen to insert the table of keys and constraints.
- *Exercise 4:* Apply Name Template to the Relational Model
  - To apply the template to the entire relational model, the first thing to do is search for the tools option and select name abbreviations. Then, I navigated to the same csv file containing the abbreviations and de-selected the table.
- *Exercise 5:* Select How Subtypes are Generated in the Relational Model
  - To define the subtypes in the relational model, I went to the logical lab and chose the faculty super type. From there, I selected properties and then subtypes from the options. Then, at subtree generation I chose the single table option and re-engineered the model.

Database Foundations 6-1: Introduction to Oracle Application Express Practices

- *Exercise 1:* Introduction to Oracle Application Express
  - Task: Go to Section 0 – Course Resources of the Learner – Learning Path for the course and access the iAcademy APEX Learner Guide. Follow the Guide to learn about the features of Oracle Application Express
    - What is APEX?
      - APEX is a web-based development environment used to build and deploy web applications. It it used with Oracle databases and has tools to help with the development process
    - What are the features in APEX?
      - APEX has features that allow for rapid development and deployment. There are interface themes, form handler, navigational controls, and more.
    - What is the architecture of APEX?
      - The architecture of APEX is meant to provide a seamless experience. It is built upon URL requests and translated through APEX PL/SQL.
    - What is the APEX environment?
      - The APEX environment is referred to as a workspace. A workspace is a private and virtual development area that allows users to collaborate while still remaining isolated with its own applications and data.

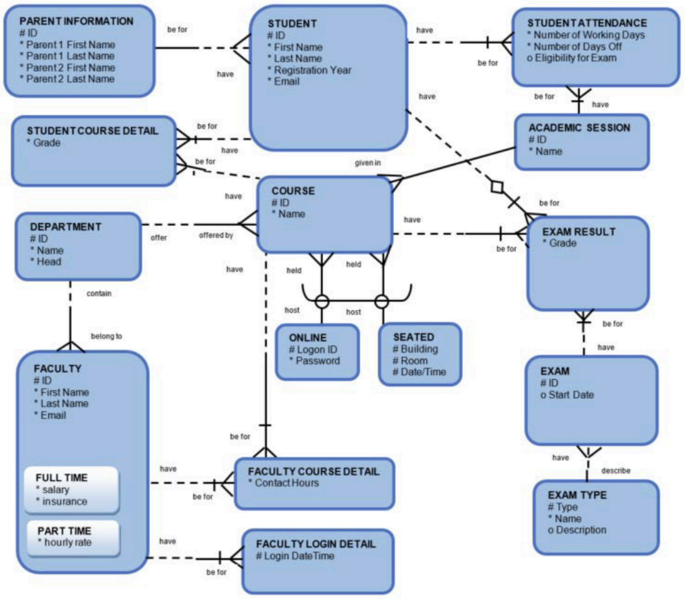
Database Foundations 6-2: Structured Query Language Practices

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

- Task: Click the Help icon, and become familiar with the following section and topics:
  - 1) Application Express SQL Workshop
    - a) Managing Database Objects with Object Browser
      - Object Browser is a tool that allows for browsing, creating, and editing of various database objects. It is divided into two sections: object selection pane and detail pane. With the tool, you have management options for many different database objects such as tables, packages, functions, indexes, and more.
    - b) Using SQL Commands
      - SQL Commands are a feature used to execute SQL statements and there is no relation with SQL Scripts. They can consist of SQL statements or PL/SQL blocks. When they are saved, SQL Commands must have a unique name in the workspace.
    - c) Using SQL Scripts
      - SQL Scripts execute sets of SQL commands that are saved in script files. They can be used to create, edit, view, run, and delete script files. You can copy and paste SQL Commands from the SQL Script to run.

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

- *Exercise 1:* Creating Tables Using Oracle Application Express
  - Assumptions: The following is the Entity Relationship Diagram (ERD) for the Academic Database where the tables will be created



- Task #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)

  -
    ```
    CREATE TABLE parent_info (
            id VARCHAR2(10) NOT NULL,
            first_name _parent1 CHAR(50) NOT NULL,
            last_name _parent1 CHAR(50) NOT NULL,
            first_name _parent2 CHAR(50) NOT NULL,
            last_name _parent2 CHAR(50) NOT NULL
            );
    ```

  -
    ```
    CREATE TABLE student (
            id VARCHAR2(10) NOT NULL,
            First_name CHAR(50) NOT NULL,
            last_name CHAR(50) NOT NULL,
            resgistration_yr NUMBER(4) NOT NULL,
            email VARCHAR2(100) NOT NULL
            );
    ```

  -
    ```
    CREATE TABLE student_attendance (
            nmbr_working_days INT NOT NULL,
            nmbr_days_off INT NOT NULL,
            exam_elgibility VARCHAR2(50)
            );
    ```

  -
    ```
    CREATE TABLE student_course_dtl (
            grade INT NOT NULL
            );
    ```

  -
    ```
    CREATE TABLE course (
            id VARCHAR2(10) NOT NULL,
            name VARCHAR2(50) NOT NULL
            );
    ```

  -
    ```
    CREATE TABLE online (
            id VARCHAR2(10) NOT NULL,
            password VARCHAR2(50) NOT NULL
            );
    ```

  -
    ```
    CREATE TABLE seated (
            building VARCHAR2(10) NOT NULL,
            room VARCHAR2(10) NOT NULL,
            date_time TIMESTAMP NOT NULL
            );
    ```

  -
    ```
    CREATE TABLE academic_session (
            id VARCHAR2(10) NOT NULL,
    ```

```
        name VARCHAR2(50) NOT NULL
        );
CREATE TABLE exam_result (
        grade INT NOT NULL
        );
CREATE TABLE exam (
        id VARCHAR2(10) NOT NULL,
        start_date DATE
        );
CREATE TABLE exam_type (
        id VARCHAR2(10) NOT NULL,
        exam_type VARCHAR2(50) NOT NULL,
        name  VARCHAR2(50) NOT NULL,
        description VARCHAR2(1000)
        );
CREATE TABLE department (
        dept_id VARCHAR2(10) NOT NULL,
        name VARCHAR2(50) NOT NULL,
        dept_head  CHAR(50)
        );
CREATE TABLE faculty (
        id VARCHAR2(10) NOT NULL,
        first_name  CHAR(50) NOT NULL,
        last_name  CHAR(50) NOT NULL,
        email VARCHAR2(100) NOT NULL
        );
CREATE TABLE faculty_ft (
        salary INT NOT NULL,
        ins_plan  VARCHAR2(50) NOT NULL
        );
CREATE TABLE faculty_pt (
        hourly_wage INT NOT NULL
        );
CREATE TABLE faculty_course_dtl (
        contact_hrs INT NOT NULL
        );
CREATE TABLE faculty_login_dtl (
        login_date_time TIMESTAMP NOT NULL
        );
```

- *Exercise 2:* Altering the Tables

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

  -
    ```
    CREATE TABLE parent_info (
            id VARCHAR2(10) NOT NULL,
            first_name _parent1 CHAR(50) NOT NULL,
            last_name _parent1 CHAR(50) NOT NULL,
            first_name _parent2 CHAR(50) NOT NULL,
            last_name _parent2 CHAR(50) NOT NULL,
            student_id VARCHAR2(10) NOT NULL,
            CONSTRAINT parent_id_pk PRIMARY KEY (id),
            CONSTRAINT student_id_fk FOREIGN KEY (student_id)
            REFERENCES student (id)
            );
    ```

  -
    ```
    CREATE TABLE student (
            id VARCHAR2(10) NOT NULL,
            First_name CHAR(50) NOT NULL,
            last_name CHAR(50) NOT NULL,
            resgistration_yr NUMBER(4) NOT NULL,
            email VARCHAR2(100) NOT NULL,
            CONSTRAINT student_id_pk PRIMARY KEY (id),
            CONSTRAINT parent_id_fk FOREIGN KEY (parent_id)
            REFERENCES parent_info (id)
            );
    ```

  -
    ```
    CREATE TABLE student_attendance (
            nmbr_working_days INT NOT NULL,
            nmbr_days_off INT NOT NULL,
            exam_elgibility VARCHAR2(50),
            CONSTRAINT student_id_uk, session_id_uk UNIQUE
            student (id), academic_session (id)
            );
    ```

-
  ```
  CREATE TABLE student_course_dtl (
          grade INT NOT NULL,
          CONSTRAINT student_id_uk, course_id_uk UNIQUE
          student (id), course (id)
          );
  ```

-
  ```
  CREATE TABLE course (
          id VARCHAR2(10) NOT NULL,
          name VARCHAR2(50) NOT NULL,
          CONSTRAINT course_id_pk PRIMARY KEY (id),
          CONSTRAINT session_id_fk, online_id_fk, seated_id_fk,
          dept_id_fk FOREIGN KEY REFERENCES academic_session
          (id), online (id), seated (id), department (id)
          );
  ```

-
  ```
  CREATE TABLE online (
          logon_id VARCHAR2(10) NOT NULL,
          password VARCHAR2(50) NOT NULL,
          CONSTRAINT logon_id PRIMARY KEY (logon id),
          CONSTRAINT course_id_fk KEY REFERENCES course (id)
          );
  ```

-
  ```
  CREATE TABLE seated (
          building VARCHAR2(10) NOT NULL,
          room VARCHAR2(10) NOT NULL,
          date_time TIMESTAMP NOT NULL,
          CONSTRAINT building_uk, room_uk, date_time_uk UNIQUE
          (building, room, date_time)
          );
  ```

-
  ```
  CREATE TABLE academic_session (
          id VARCHAR2(10) NOT NULL,
          name VARCHAR2(50) NOT NULL,
          CONSTRAINT session_id_pk PRIMARY KEY (id),
          CONSTRAINT student_id_fk FOREIGN KEY REFERENCES
          student (id)
          );
  ```

-
  ```
  CREATE TABLE exam_result (
          grade INT NOT NULL,
          CONSTRAINT student_id_uk, exam_id_uk, course_id_uk
          UNIQUE student (id), exam (id), course (id)
          );
  ```

-
  ```
  CREATE TABLE exam (
          id VARCHAR2(10) NOT NULL,
  ```

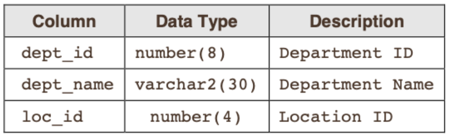
```
            start_date DATE,
            course_id VARCHAR2(10) NOT NULL,
            CONSTRAINT exam_id_pk PRIMARY KEY (id),
            CONSTRAINT course_id FOREIGN KEY REFERENCES course
            (id)
            );
■   CREATE TABLE exam_type (
            type VARCHAR2(50) NOT NULL,
            name  VARCHAR2(50) NOT NULL,
            description VARCHAR2(1000),
            CONSTRAINT exam_type_pk PRIMARY KEY (type),
            CONSTRAINT exam_id_fk FOREIGN KEY REFERENCES
            exam (id)
            );
■   CREATE TABLE department (
            id VARCHAR2(10) NOT NULL,
            name VARCHAR2(50) NOT NULL,
            dept_head  CHAR(50),
            CONSTRAINT dept_id_pk PRIMARY KEY (id)
            );
■   CREATE TABLE faculty (
            id VARCHAR2(10) NOT NULL,
            first_name  CHAR(50) NOT NULL,
            last_name  CHAR(50) NOT NULL,
            email VARCHAR2(100) NOT NULL,
            full_time_id NUMBER(10),
            part_time_id NUMBER(10),
            CONSTRAINT faculty_id_pk PRIMARY KEY (id),
            CONSTRAINT full_time_id_fk, part_time_id_fk FOREIGN KEY
            REFERENCES faculty_ft (id), faculty_pt (id)
            );
■   CREATE TABLE faculty_ft (
            id VARCHAR2(10) NOT NULL,
            salary INT NOT NULL,
            ins_plan  VARCHAR2(50) NOT NULL,
            CONSTRAINT full_time_id_pk PRIMARY KEY (id),
            CONSTRAINT faculty_id_fk FOREIGN KEY REFERENCES
            faculty (id)
            );
■   CREATE TABLE faculty_pt (
id VARCHAR2(10) NOT NULL,
hourly_wage INT NOT NULL,
CONSTRAINT part_time_id_pk PRIMARY KEY (id),
CONSTRAINT faculty_id_fk FOREIGN KEY REFERENCES
faculty (id)
);
```

        -
          ```
          CREATE TABLE faculty_course_dtl (
                  id VARCHAR2(10) NOT NULL,
                  contact_hrs INT NOT NULL,
                  faculty_id VARCHAR2(10) NOT NULL,
                  course_id VARCHAR2(10) NOT NULL,
                  CONSTRAINT faculty_course_id_pk  PRIMARY KEY (id),
                  CONSTRAINT faculty_id_fk, course_id_fk FOREIGN KEY
                  REFERENCES faculty (id), course (id)
                  );
          ```

        -
          ```
          CREATE TABLE faculty_login_dtl (
                  login_date_time TIMESTAMP NOT NULL,
                  CONSTRAINT login_date_time_pk PRIMARY KEY
                  (login_date_time),
                  CONSTRAINT faculty_id_fk FOREIGN KEY REFERENCES
                  faculty (id)
                  );
          ```
    - Task #2: Alter the table AD_FACULTY_LOGIN_DETAILS and specify a default value for the column LOGIN_DATE_TIME of SYSDATE

        -
          ```
          ALTER TABLE AD_FACULTY_LOGIN_DETAILS
          MODIFY LOGIN_DATE_TIME SYSDATE NOT NULL
          ```
    - Task #3: Set the AD_PARENT_INFORMATION table to a read-only status

        -
          ```
          ALTER TABLE PARENT_INFORMATION READ ONLY
          ```
- *Exercise 3:* Creating Composite Primary, Foreign and Unique Keys
    - Task #1: The primary key for this table needs to be defined as a composite comprising of the dept_id and loc_id. Create the DEPT table with the following structure:

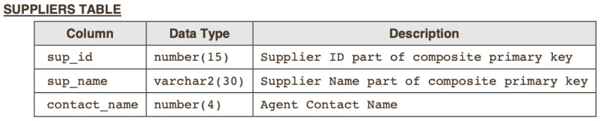


Column	Data Type	Description
dept_id	number(8)	Department ID
dept_name	varchar2(30)	Department Name
loc_id	number(4)	Location ID

        -
          ```
          CREATE TABLE dept (
          dept_id NUMBER(8),
          dept_name VARCHAR2(30),
          loc_id NUMBER(4),
          CONSTRAINT dept_id_pk, loc_id_pk PRIMARY (dept_id, loc_id)
          ```

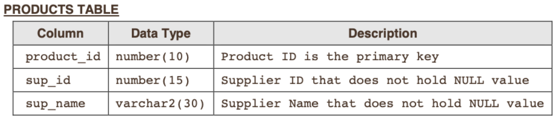
);

- Task #2: The primary key for this table needs to be defined as a composite comprising of the sup_id and sup_name. The primary key for this table is product_id. The foreign key for this table needs to be defined as a composite comprising of the sup_id and sup_name. Create the SUPPLIERS and PRODUCTS table with the following structure:

  SUPPLIERS TABLE

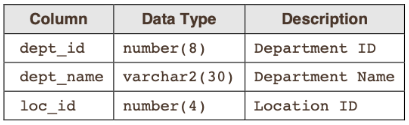
Column	Data Type	Description
sup_id	number(15)	Supplier ID part of composite primary key
sup_name	varchar2(30)	Supplier Name part of composite primary key
contact_name	number(4)	Agent Contact Name

  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

  - CREATE TABLE suppliers (
    sup_id NUMBER(15),
    sup_name VARCHAR2(30),
    contact_name NUMBER(4),
    CONSTRAINT sup_id_uk, sup_name_uk PRIMARY (sup_id, sup_name)
    );
  - CREATE TABLE products (
    product_id NUMBER(10),
    sup_id NUMBER(15),
    sup_name VARCHAR2(30),
    CONSTRAINT product_id_pk PRIMARY KEY (product_id),
    CONSTRAINT sup_id_fk, sup_name_fk FOREIGN KEY
    REFERENCES suppliers (sup_id, sup_name)
    );
- Task #3: The UNIQUE key for this table needs to be defined as a composite comprising of the dept_id and dept_name. Create the DEPT_SAMPLE table with the following structure:

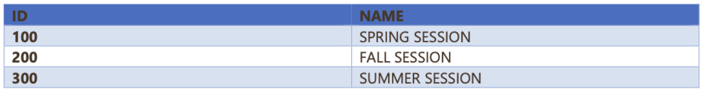
Column	Data Type	Description
dept_id	number(8)	Department ID
dept_name	varchar2(30)	Department Name
loc_id	number(4)	Location ID

  - CREATE TABLE dept_sample (
    dept_id NUMBER(8),
    dept_name VARCHAR2(30),
    loc_id NUMBER(4),
    CONSTRAINT dept_id_uk , dept_name_uk UNIQUE (dept_id, dept_name)
    );

Database Foundations 6-4: Defining Data Manipulation Practices

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

AD_ACADEMIC_SESSIONS:

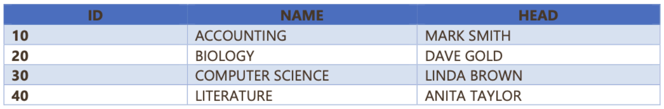


ID	NAME
100	SPRING SESSION
200	FALL SESSION
300	SUMMER SESSION

■

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

AD_DEPARTMENTS:

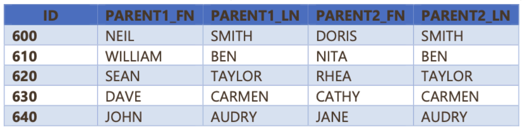


ID	NAME	HEAD
10	ACCOUNTING	MARK SMITH
20	BIOLOGY	DAVE GOLD
30	COMPUTER SCIENCE	LINDA BROWN
40	LITERATURE	ANITA TAYLOR

■

```
INSERT INTO AD_DEPARTMENTS (ID, NAME, HEAD)
VALUES (10, ‘ACCOUNTING’, ‘MARK_SMITH’),
            (20, ‘BIOLOGY’, ‘DAVE_GOLD’),
            (30, ‘COMPUTER_SCIENCE’, ‘LINDA_BROWN’),
            (40, ‘SUMMER SESSION’, ‘ANITA_TAYLOR’);
```

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

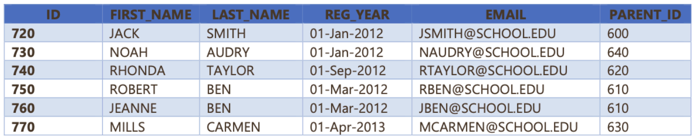


ID	PARENT1_FN	PARENT1_LN	PARENT2_FN	PARENT2_LN
600	NEIL	SMITH	DORIS	SMITH
610	WILLIAM	BEN	NITA	BEN
620	SEAN	TAYLOR	RHEA	TAYLOR
630	DAVE	CARMEN	CATHY	CARMEN
640	JOHN	AUDRY	JANE	AUDRY

■

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

AD_STUDENTS:

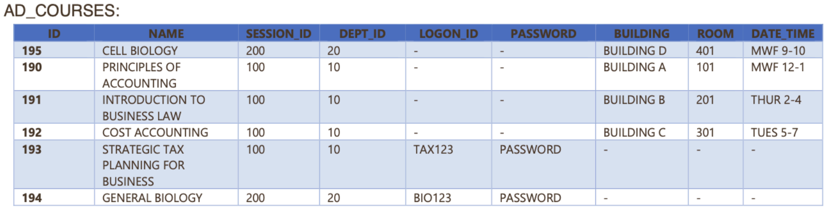


ID	FIRST_NAME	LAST_NAME	REG_YEAR	EMAIL	PARENT_ID
720	JACK	SMITH	01-Jan-2012	JSMITH@SCHOOL.EDU	600
730	NOAH	AUDRY	01-Jan-2012	NAUDRY@SCHOOL.EDU	640
740	RHONDA	TAYLOR	01-Sep-2012	RTAYLOR@SCHOOL.EDU	620
750	ROBERT	BEN	01-Mar-2012	RBEN@SCHOOL.EDU	610
760	JEANNE	BEN	01-Mar-2012	JBEN@SCHOOL.EDU	610
770	MILLS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

■

```
INSERT INTO AD_STUDENTS (FIRST_NAME, LAST_NAME,
REG_YEAR, EMAIL, PARENT_ID)
VALUES (720, ‘JACK’, ‘SMITH’, ‘01-Jan-2012’,
            ‘JSMITH@SCHOOL.EDU’, ‘600’),
            (730, ‘NOAH’, ‘AUDRY’, ‘01-Jan-2012’,
```

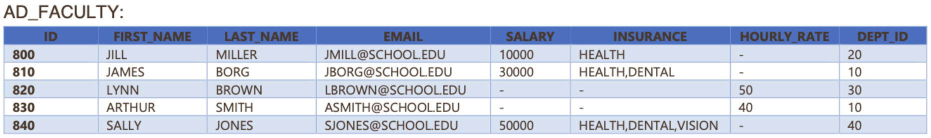
‘NAUDRY@SCHOOL.EDU’, ‘640’),
(740, ‘RHONDA’, ‘TAYLOR’, ‘01-Sep-2012’,
‘RTAYLOR@SCHOOL.EDU’, ‘620’),
(750, ‘ROBERT’, ‘BEN’, ‘01-Mar-2012’,
‘RBEN@SCHOOL.EDU’, ‘610’),
(760, ‘JEANNE’, ‘BEN’, ‘01-Mar-2012’,
‘JBEN@SCHOOL.EDU’, ‘610’),
(770, ‘MILLS’, ‘CARMEN’, ‘01-Apr-2013’,
‘MCARMEN@SCHOOL.EDU’, ‘630’);



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	-	-	-

- INSERT INTO AD_COURSES (ID, NAME, SESSION_ID, DEPT_ID,
LOGON_ID, PASSWORD, BUILDING, ROOM, DATE_TIME)
VALUES (195, ‘CELL_BIOLOGY’, 200, 20, NULL, NULL,
‘BUILDING_D’, 401, ‘MWF_9-10’),
(190, ‘PRINCIPLES_OF_ACCOUNTING’, 100, 10, NULL,
NULL, ‘BUILDING_A’, 101, ‘MWF_12-1’),
(191, ‘INTRODUCTION_TO_BUSINESS_LAW’, 100, 10,
NULL, NULL, ‘BUILDING_B’, 201, ‘THUR_2-4’),
(192, ‘COST_ACCOUNTING’, 100, 10, NULL, NULL,
‘BUILDING_C’, 301, ‘TUES_5-7’),
(193, ‘STRATEGIC_TAX_PLANNING_FOR_BUSINESS’,
100, 10, NULL, ‘TAX123’, ‘PASSWORD’, NULL, NULL,
NULL),
(194, ‘GENERAL_BIOLOGY’, 200, 20, ‘BIO123’,
‘PASSWORD’, NULL, NULL, NULL);



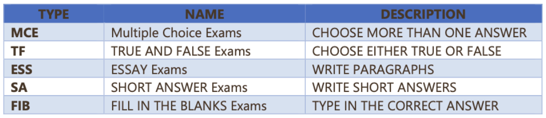
AD_FACULTY:

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

- INSERT INTO AD_FACULTY (ID, FIRST_NAME, LAST_NAME,
EMAIL, SALARY, INSURANCE, HOURLY_RATE, DEPT_ID)
VALUES (800, ‘JILL’, ‘MILLER’, ‘JMILL@SCHOOL.EDU’, 10000,
‘HEALTH’, NULL, 20),
(810, ‘JAMES’, ‘BORG’, ‘JBORG@SCHOOL.EDU’, 30000,
‘HEALTH,DENTAL’, NULL, 10),
(820, ‘LYNN’, ‘BROWN’, ‘LBROWN@SCHOOL.EDU’,

NULL, NULL, 50, 30),
(830, ‘ARTHUR’, ‘SMITH’, ‘ASMITH@SCHOOL.EDU’,
NULL, NULL, 40, 10),
(840, ‘SALLY’, ‘JONES’, ‘SJONES@SCHOOL.EDU’, 50000,
‘HEALTH,DENTAL,VISION’, NULL, 40);

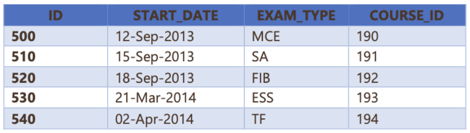
AD_EXAM_TYPES:



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 (‘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’)

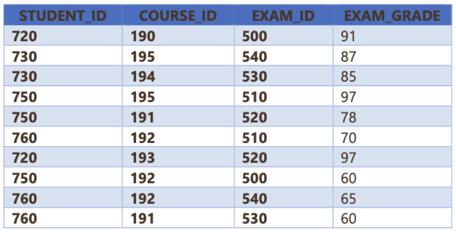
AD_EXAMS:



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 AD_EXAMS (ID, START_DATE, EXAM_TYPE, COURSE_ID)
VALUES (500, ‘12-Sep-2013’, ‘MCE’, 190),
(510, ‘15-Sep-2013’, ‘SA’, 191),
(520, ‘18-Sep-2013’, ‘FIB;, 192),
(530, ‘21- Mar-2014’, ‘ESS’, 193),
(540, ‘02-Apr-2014’, ‘TF’, 194);

AD_EXAM_RESULTS:

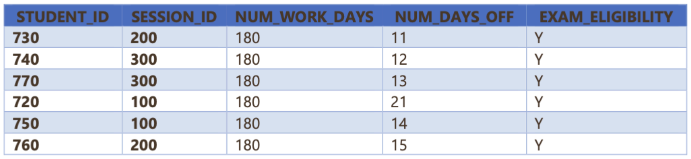


STUDENT_ID	COURSE_ID	EXAM_ID	EXAM_GRADE
720	190	500	91
730	195	540	87
730	194	530	85
750	195	510	97
750	191	520	78
760	192	510	70
720	193	520	97
750	192	500	60
760	192	540	65
760	191	530	60

- INSERT INTO AD_EXAMS_RESULTS (STUDENT_ID, COURSE_ID, EXAM_ID, EXAM_GRADE)
VALUES (720, 190, 500, 91),

(730, 195, 540, 87),
(730, 194, 530, 85),
(750, 195, 510, 97),
(750, 191, 520, 78),
(760, 192, 510, 70),
(720, 193, 520, 97),
(750, 192, 500, 60),
(760, 192, 540, 65),
(760, 191, 530, 60);

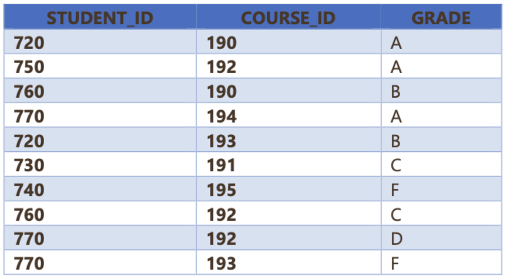
AD_STUDENT_ATTENDANCE:



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_ATTENDACE (STUDENT_ID, SESSION_ID, NUM_WORK_DAYS, NUM_DAYS_OFF, EXAM_ELIGIBILITY)
  VALUES (730, 200, 180, 11, ‘Y’),
  (740, 300, 180, 12 ‘Y’),
  (770, 300, 180, 13, ‘Y’),
  (720, 100, 180, 21, ‘Y’),
  (750, 100, 180, 14 ‘Y’),
  (760, 200, 180, 15, ‘Y’);

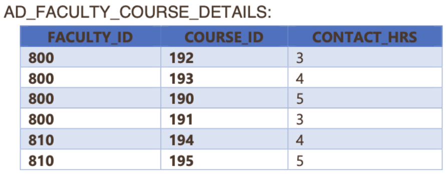
AD_STUDENT_COURSE_DETAILS:



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

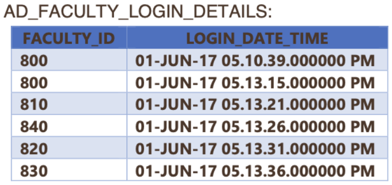
- INSERT INTO AD_STUDENT_COURSE_DETAILS (STUDENT_ID, COURSE_ID, GRADE)
  VALUES (720, 190, ‘A’)
  (750, 192, ‘A’)
  (760, 190, ‘B’)
  (770, 194, ‘A’)
  (720, 193, ‘B’)
  (730, 191, ‘C’)
  (740, 195, ‘F’)
  (760, 192, ‘C’)

(770, 192, ‘D’)
(770, 193, ‘F’)


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 (800, 192, 3)
  (800, 193, 4)
  (800, 190, 5)
  (800, 191, 3)
  (810, 194, 4)
  (810, 195, 5)


AD_FACULTY_LOGIN_DETAILS:

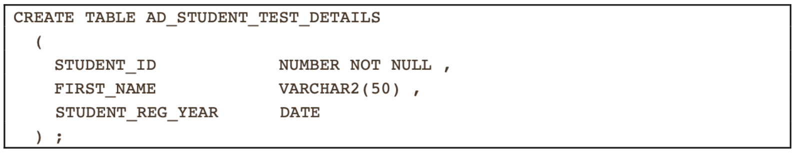
FACULTY_ID	LOGIN_DATE_TIME
800	01-JUN-17 05.10.39.000000 PM
800	01-JUN-17 05.13.15.000000 PM
810	01-JUN-17 05.13.21.000000 PM
840	01-JUN-17 05.13.26.000000 PM
820	01-JUN-17 05.13.31.000000 PM
830	01-JUN-17 05.13.36.000000 PM

- INSERT INTO AD_FACULTY_LOGIN_DETAILS (FACULTY_ID, LOGIN_DATE_TIME)
  VALUES (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’);

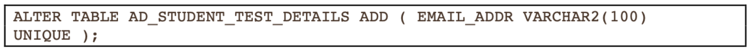
- *Exercise 2:* Updating Rows in the Tables
  - Task #1: Alter the AD_FACULTY_LOGIN_DETAILS table to add a field called DETAILS make it a VARCHAR2(50) character field – it can have null values.
    - ALTER TABLE AD_FACULTY_LOGIN_DETAILS
      ADD DETAILS VARCHAR2(50);
  - Task #2: Update at least 2 records in the DETAILS column in the faculty login details table.
    - UPDATE AD_FACULTY_LOGIN_DETAILS
      SET DETAILS = ‘NOT_UPDATED’
      WHERE ID = 1;
    - UPDATE AD_FACULTY_LOGIN_DETAILS
      SET DETAILS = ‘UPDATED’
      WHERE ID = 2;

Database Foundations 6-5: Defining Transaction Control Practices

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

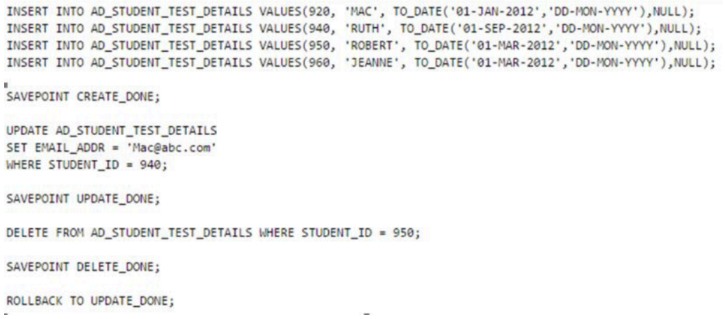


```
CREATE TABLE AD_STUDENT_TEST_DETAILS
  (
    STUDENT_ID            NUMBER NOT NULL ,
    FIRST_NAME            VARCHAR2(50) ,
    STUDENT_REG_YEAR      DATE
  ) ;
```



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

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



```
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;
```

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

Database Foundations 6-6: Retrieving Data Practices

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

```
FROM AD_PARENT_INFORMATION;
```

    - Task #2: Write a query to retrieve the exam grade obtained by each student for every exam attempted

        -
          ```
          SELECT *
          FROM AD_EXAMS_RESULTS;
          ```
    - Task #3: Write a query to check if a student is eligible to take exams based on the number of days he/she attended classes

        -
          ```
          SELECT *
          FROM AD_STUDENT_ATTENDACE;
          ```
    - Task #4: Display the LOGIN_DATE_TIME for each faculty member

        -
          ```
          SELECT LOGIN_DATE_TIME
          FROM AD_FACULTY_LOGIN_DETAILS;
          ```
    - Task #5: Display the name of the Head of the Department for each of the Departments

        -
          ```
          SELECT HEAD
          FROM AD_DEPARTMENTS;
          ```
    - Task #6: Retrieve the student ID and first name for each student concatenated with literal text to look like this: 720: FIRST NAME IS JACK

        -
          ```
          SELECT STUDENT_ID || ‘: FIRST NAME IS’ || FIRST_NAME AS
          STUDENT_INFORMATION
          FROM AD_STUDENTS;
          ```
    - Task #7: Display all the distinct exam types from the AD_EXAMS table

        -
          ```
          SELECT DISTINCT TYPE
          FROM AD_EXAMS;
          ```

Database Foundations 6-7: Restricting Data Using SELECT

- *Exercise 1:* Restricting Data Using SELECT
    - Task #1: Display the course details for the Spring Session

        -
          ```
          SELECT *
          FROM AD_COURSES
          WHERE SESSION_ID = 100;
          ```
    - Task #2: Display the details of the students who have scored more than 95

        -
          ```
          SELECT *
          FROM AD_EXAM_RESULTS
          WHERE GRADE > 95;
          ```
    - Task #3: Display the details of the students who have scored between 65 and 70

        -
          ```
          SELECT *
          FROM AD_EXAM_RESULTS
          WHERE GRADE BETWEEN 65 AND 70;
          ```
    - Task #4: Display the students who registered after 01-Jun-2012

        -
          ```
          SELECT *
          ```

```
        FROM AD_STUDENTS
        WHERE REG_YEAR > ‘01-JUN-2012’;
        ```
    - Task #5: Display the course details for departments 10 and 30

        -
          ```
          SELECT *
          FROM AD_COURSES
          WHERE DEPT_ID IN (10, 30);
          ```
    - Task #6: Display the details of students whose first name begins with the letter "J"

        -
          ```
          SELECT *
          FROM AD_STUDENTS
          WHERE FIRST_NAME LIKE ‘J%’;
          ```
    - Task #7: Display the details of students who have opted for courses 190 or 193

        -
          ```
          SELECT *
          FROM AD_STUDENT_COURSE_DETAILS
          WHERE COURSE_ID IN (190, 193);
          ```
    - Task #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;
          ```
    - Task #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);
          ```
    - Task #10: Display the course details for department 2

        -
          ```
          SELECT *
          FROM AD_COURSES
          WHERE DEPT_ID = 20;
          ```

Database Foundations 6-8: Sorting Data Using ORDER BY Practices

- *Exercise 1:* Sorting Data Using ORDER BY
    - Task #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 ASC;
              ```
        - b. AD_EXAM_RESULTS ordered by STUDENT_ID and COURSE_ID

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

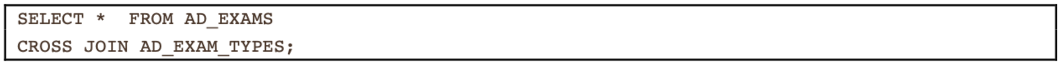
        - c. AD_STUDENT_ATTENDANCE ordered by STUDENT_ID
            - SELECT *
              FROM AD_STUDENT_ATTENDANCE
              ORDER BY STUDENT_ID ASC;
        - d. AD_DEPARTMENTS ordered by the department ID
            - SELECT *
              FROM AD_DEPARTMENTS
              ORDER BY DEPARTMENT_ID ASC;
    - Task #2: Display the percentage of days students have taken days off and sort the records based on the percentage calculated
        - SELECT STUDENT_ID, (NUM_DAYS_OFF / NUM_WORK_DAYS) * 100 AS ABSENCE_PERCENTAGE
          FROM AD_STUDENT_ATTENDANCE
          ORDER BY ABSENCE_PERCENTAGE DESC;
    - Task #3: Display the top 5 students based on exam grade results
        - SELECT STUDENT_ID, GRADE
          FROM AD_EXAM_RESULTS
          ORDER BY GRADE DESC
          LIMIT 5;
    - Task #4: Display the parent details ordered by the parent ID
        - SELECT *
          FROM AD_PARENTS
          ORDER BY PARENT_ID ASC;

Database Foundations 6-9: Joining Tables Using JOIN Practices

- *Exercise 1:* Using JOINS in SQL Queries
    - Task #1: Display the different courses offered by the departments in the school
        - SELECT C.COURSE_NAME, D.DEPT_NAME
          FROM AD_COURSES C
          JOIN AD_DEPARTMENTS D ON C.DEPARTMENT_ID = D.DEPARTMENT_ID;
    - Task #2: Display the courses offered in the Fall session
        - SELECT COURSE_NAME
          FROM AD_COURSES
          WHERE SESSION_ID = 200;
    - Task #3: Display the course details, the department that offers the courses and students who have enrolled for those courses
        - SELECT C.COURSE_NAME, D.DEPT_NAME, S.STUDENT_NAME
          FROM AD_COURSES C
          JOIN AD_DEPARTMENTS D ON C.DEPARTMENT_ID = D.DEPARTMENT_ID

JOIN AD_ENROLLMENTS E ON C.COURSE_ID = E.COURSE_ID
JOIN AD_STUDENTS S ON E.STUDENT_ID = S.STUDENT_ID;

- Task #4: Display the course details, the department that offers the courses and students who have enrolled for those courses for department 20
    - SELECT C.COURSE_NAME, D.DEPT_NAME, S.STUDENT_NAME
      FROM AD_COURSES C
      JOIN AD_DEPARTMENTS D ON C.DEPARTMENT_ID = D.DEPARTMENT_ID
      JOIN AD_ENROLLMENTS E ON C.COURSE_ID = E.COURSE_ID
      JOIN AD_STUDENTS S ON E.STUDENT_ID = S.STUDENT_ID
      WHERE D.DEPARTMENT_ID = 20;
- Task #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 S.STUDENT_NAME, .GRADE
      FROM AD_EXAM_RESULTS E
      JOIN AD_STUDENTS S ON E.DEPARTMENT_ID = S.STUDENT_ID
      WHERE E.COURSE_ID BETWEEN 190 AND 192;
- Task #6: Retrieve the rows from the AD_EXAM_RESULTS table even if there are no matching records in the AD_COURSES table
    - SELECT E.*, C.COURSE_NAME
      FROM AD_EXAM_RESULTS E
      LEFT JOIN AD_COURSES C ON E.COURSE_ID = C.COURSE_ID
- Task #7: What output would be generated when the given statement is executed



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

    - The output would be the combination of every row from AD_EXAMS with every row from AD_EXAM_TYPES