**Database Foundations  
5-1: Mapping Entities and Attributes  
Practices**

A screenshot of a computer

Description automatically generated

* Using the Oracle SQL Developer Data Modeler I created a Glossary from the Logical Model.
* This action will extract the entities, attributes, and relationships defined in the Logical Model to create a glossary. The glossary typically contains definitions of terms, attributes, and classifications used in the model.
* I then gave the glossary a name (e.g., “School Management System) and a brief description.
* Next, I specified classification types including entities and attributes. The next step once the glossary was created was to save the glossary.

A screenshot of a computer

Description automatically generated

* To ensure the glossary is applied when forward engineering the model in Oracle SQL Developer Data Modeler, the following steps were taken:
  + Right-clicked Design model in the Browser and selected Properties.
  + Expanded Settings and clicked on the Naming Standard node.
  + Added the glossary by clicking + in the Glossary region and navigating to the glossary file.
  + Clicked the Engineer (>>) icon to open the Engineering to Relational Model dialogue.
  + In the General Options tab, check Apply name translation.
  + Ensured that the use of preferred abbreviations was selected.
  + Clicked Engineer to apply the glossary and naming standards during forward engineering.
* This process helps ensure consistency in naming and terminology when transitioning from the Logical Model to the Relational Model.

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

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

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

**Primary Keys (Unique Identifiers mapped as Primary Keys):**

* **Logical Model**:
  + The Logical Model represents primary keys with a # symbol.
  + In the Logical Model:
    - Course ID is the primary key for the Course entity.
    - Student ID is the primary key for the student entity.
    - ID is the primary key for the Parent Information entity.
* **Relational Model**:
  + The engineered Relational Model shows primary keys marked with a P.
  + In the Relational Model:
    - Course ID is marked as the primary key for Course.
    - Student ID is marked as the primary key for Student.
    - ID is marked as the primary key for Parent\_Information.

**2. Unique Keys (Mapped as Unique Keys):**

* **Logical Model**:
  + The Logical Model does not explicitly indicate any unique keys.
* **Relational Model**:
  + The engineered Relational Model does not indicate any additional fields as unique keys (only primary keys are highlighted).
* **Conclusion**: There are no unique keys defined explicitly in either model other than the primary keys.

**3. Foreign Keys (Relationships mapped as Foreign Keys):**

* **Logical Model**:
  + The Logical Model represents relationships between the entities through the arrows connecting them.
  + Student has a relationship with Course (implying Course ID as a foreign key in Student).
  + Parent\_Information is related to Student (implying Student ID as a foreign key in Parent Information).
* **Relational Model**:
  + In the Relational Model, foreign keys are marked with F symbols.
  + In the Relational Model:
    - Course ID in the Student table is marked as a foreign key referencing Course ID.
    - Student ID in the Parent\_Information table is marked as a foreign key referencing Student ID.

A diagram of a student

Description automatically generatedA diagram of a student

Description automatically generated

(Relational Model) (Logical Model)

Exercise 2: Define table name abbreviations in a .csv file

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

A screenshot of a computer

Description automatically generated

Exercise 3: Define Name Template

You can set a template for the keys, indexes, and constraints in a 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:

* A screenshot of a computer

  Description automatically generatedBelow I have attached a screenshot of the template I set for keys, indexes, and constraints in the Oracle SQL Developer Data Modeler using a combination of predefined variables to define name patterns.

Exercise 4: Apply Name Template to the Relational Model

Tasks  
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

A screenshot of a computer

Description automatically generated

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

Tasks  
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.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

**Faculty Table (Single Table)**:

* The faculty table now contains all attributes that would have been part of both the Member subtype and the faculty supertype.
* The attributes ID, First Name, Last Name, Street Address, City, State, and Zip, which were part of the Member subtype, are now included in the faculty table.
* The **Details** attribute, specific to the faculty subtype, is now also part of the same table.
* There is no longer a separate **Member** table, which means this approach combines everything into one table, reducing the need for foreign keys and separate connections across tables.

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

Exercise 1: Introduction to Oracle Application Express

Tasks  
1. Go to Section 0 – Course Resources of the Learner – Learning Path for the course and access the iAcademy APEX Learner  
Guide.  
2. Follow the Guide to learn about the features of Oracle Application Express.

**Features of Oracle Application Express:**

* **Low-code Development**: Allows developers to build applications quickly using minimal coding.
* **Web-based Interface**: Provides an intuitive web interface to develop applications without needing complex software installation.
* **Workspaces**: Isolated development areas where you can manage multiple applications and database schemas.
* **SQL Workshop**: Tools for running SQL commands, managing databases, and building data models.
* **Application Builder**: A graphical environment to create pages, forms, reports, and dashboards easily.
* **Interactive Grids and Reports**: APEX supports interactive grids, allowing users to view, search, and manipulate data in real-time.
* **Data Management**: Easy integration with databases to store and manage large volumes of data.
* **Security**: Built-in mechanisms for user authentication, data protection, and access control within applications.
* **Mobile Compatibility**: Responsive designs for applications to work across different devices, including mobile and tablets.
* **APIs and Integration**: Supports integration with REST APIs and other external systems.
* **Deployment and Hosting**: Simplified options for deploying applications to Oracle Cloud or other hosting environments.
* **Customization**: Flexible theming and customization options to control the appearance and functionality of applications.

**Database Foundations  
6-2 : Structured Query Language  
Practices**

Exercise 1: Using Help in Oracle Application Express

Tasks  
1. Access and log in to Oracle Application Express

2. 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  
 b) Using SQL Commands  
 c) Using SQL Scripts

**1) Managing Database Objects with Object Browser:**

* View and manage tables, views, sequences, and indexes.
* Create and modify database objects.
* View and edit data within tables.

**2) Using SQL Commands:**

* Execute SQL queries and commands.
* Access previously executed commands via history.

**3) Using SQL Scripts:**

* Create and run multiple SQL statements in sequence.
* Save, manage, and reuse SQL scripts.
* View script execution results and error messages.

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

Exercise 1: Creating Tables Using Oracle Application Express

Tasks  
1. Create the DDL Statements for creating the tables for the Academic Database listed above – include NOT NULL constraints  
where necessary. (Other constraints will be added later)  
2. Run/execute these commands in Oracle Application Express

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Exercise 2: Altering the Tables  
Tasks  
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.  
NOTE: You can execute the INSERT / ALTER TABLE statements in Oracle Application Express in one of the two ways:  
Method 1:  
a. Open Oracle Application Express and paste the commands into the SQL Commands screen one at a time and run.  
Method 2:  
a. Open Oracle Application Express and use the same script upload method as you did with the DDL commands above.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

**Summary of Exercise:**

* **Created and Altered Tables:**
  + Generated DDL (Data Definition Language) statements to create various tables, including COURSES, STUDENTS, FACULTY, and others, with NOT NULL constraints.
  + Modified tables to add primary keys, foreign keys, and unique constraints to ensure data integrity across the database.
* **Checked Existing Constraints:**
  + Used the user\_constraints and user\_cons\_columns commands to identify existing constraints on tables, including primary keys (P), foreign keys (R), and check constraints (C).
  + Identified that certain constraints, such as foreign keys, already existed, preventing redefinition.
* **Added and Verified Foreign Key Constraints:**
  + Successfully added foreign key constraints to tables such as FACULTY\_COURSE\_DETAIL, ensuring proper relationships between tables (e.g., Faculty\_ID referencing FACULTY).
* **Set Default Value for a Column:**
  + Altered the FACULTY\_LOGIN\_DETAIL table to set a default value (SYSDATE) for the LOGINDATETIME column, ensuring that the current system date and time are automatically populated upon insertion.
* **Made a Table Read-Only:**
  + Altered the PARENT\_INFORMATION table to be read-only, preventing any inserts, updates, or deletes, ensuring the data remains static.
* **Learned Key Oracle SQL Commands:**
  + Learned how to use the ALTER TABLE statement to modify tables and add constraints like primary keys, foreign keys, and defaults.

Exercise 3: Creating Composite Primary, Foreign and Unique Keys

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer program

Description automatically generated

A screenshot of a computer

Description automatically generated

* **Created DEPT Table**: A table named DEPT was created with a composite primary key combining dept\_id and loc\_id, ensuring the unique identification of each department by both columns.
* **Created SUPPLIERS Table**: The SUPPLIERS table was created with a composite primary key comprising sup\_id and sup\_name, which ensures that each supplier is uniquely identified by both the supplier ID and name.
* **Created PRODUCTS Table**: The PRODUCTS table was created with a foreign key relationship to the SUPPLIERS table, where the combination of sup\_id and sup\_name references the composite primary key in the SUPPLIERS table. Additionally, product\_id was defined as the primary key for this table.
* **Created DEPT\_SAMPLE Table**: The DEPT\_SAMPLE table was created with a unique composite key combining dept\_id and dept\_name, enforcing uniqueness on these two columns.
* **Used Composite Keys**: For tables like DEPT, SUPPLIERS, and DEPT\_SAMPLE, composite keys were employed to ensure unique identification based on multiple columns.

**Database Foundations  
6-4: Defining Data Manipulation  
Practices**

Exercise 1: Inserting Rows in Tables

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

**Inserting Data into Various Tables**:

* Data was inserted into multiple tables in the database, including ACADEMIC\_SESSION, DEPARTMENT, PARENT\_INFORMATION, STUDENT, COURSE, FACULTY, and STUDENT\_ATTENDANCE,
* Each table had corresponding INSERT statements aligned with the data structure and constraints of the respective table.

**Addressing Data Constraints**:

* Specific constraints such as valid values for columns (e.g., YES/NO for ELIGIBILITYFOREXAM) were considered when inserting data, ensuring that constraints were respected.

**Resolving Errors Step-by-Step**:

* Errors encountered during the process, such as mismatched column names or invalid data values, were resolved by checking the schema and aligning the INSERT statements with the correct data format and constraints.

Exercise 2: Updating Rows in the Tables

You will update the records in FACULTY\_LOGIN\_DETAILS table to include a DETAILS field in the table.  
Tasks  
1. Alter the AD\_FACULTY\_LOGIN\_DETAILS table to add a field called DETAILS make it a VARCHAR2(50) character field – it can  
have null values.  
2. Update at least 2 records in the DETAILS column in the faculty login details table.  
\*\*Note: You will have to look up the LOGIN\_DATE\_TIME values for the records being updated since it is  
part of the primary key.  
Verify that the DETAILS column has been updated with the values:

A screenshot of a computer program

Description automatically generated

A screenshot of a computer

Description automatically generated

**Altered Table Structure**: Added a new column DETAILS (of type VARCHAR2(50)) to the FACULTY\_LOGIN\_DETAIL table, allowing null values.

**Updated Records**: Updated two records in the FACULTY\_LOGIN\_DETAIL table, adding specific values to the newly created DETAILS column based on the FACULTY\_ID and LOGINDATETIME.

**Verified Data Update**: Queried the table to confirm that the new DETAILS values were successfully added to the specified rows.

**Database Foundations  
6-5: Defining Transaction Control  
Practices**

Exercise 1: Controlling Transactions

**Table Creation and Modification**:

* A new table (AD\_STUDENT\_TEST\_DETAILS) was created with columns for student ID, first name, and registration year.
* The table was altered to add an email address column, demonstrating the use of the ALTER TABLE command.

**Save points**:

* A save point (ALTER\_DONE) was created after the table alteration to mark a point in the transaction for potential rollback.
* Save points allow you to rollback to a specific point in the transaction without affecting prior changes.

**Rollback Behavior**:

* Rolling back to ALTER\_DONE would not remove the EMAIL\_ADDR column since structural changes from ALTER TABLE are not undone by rollback operations.

**Data Manipulation**:

* Multiple records were inserted into the table, followed by save points to track changes at various stages (INSERT\_DONE, UPDATE\_DONE, DELETE\_DONE).

**Rollback Implications**:

* Rolling back to UPDATE\_DONE undoes changes made after that save point, restoring deleted records while retaining updates made before the save point.
* This showcases how rollback operations can selectively undo changes while preserving others.

**Understanding Transaction Control**:

* The exercise emphasized the importance of TCL commands in managing database transactions, ensuring data integrity, and controlling changes made during a session.

**Database Foundations  
6-6: Retrieving Data  
Practices**

Exercise 1: Retrieving Columns from tables

Tasks  
1. Write a simple query to view the data inserted in the tables created for the academic database  
2. Write a query to retrieve the exam grade obtained by each student for every exam attempted.  
3. Write a query to check if a student is eligible to take exams based on the number of days he/she attended classes.  
4. Display the LOGIN\_DATE\_TIME for each faculty member.  
5. Display the name of the Head of the Department for each of the 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 screenshot of a computer

Description automatically generated7. Display all the distinct exam types from the AD\_EXAMS table.

* 1. simple query to view the data inserted in the tables created for the academic database

A screenshot of a computer

Description automatically generated

* 1. Write a query to retrieve the exam grade obtained by each student for every exam attempted.

A screenshot of a computer

Description automatically generated

* 1. A student is eligible to take exams based on the number of days he/she attended classes.

A screenshot of a computer

Description automatically generated

* 1. Display the LOGIN\_DATE\_TIME for each faculty member.

A screenshot of a computer

Description automatically generated

* 1. Display the name of the Head of the Department for each of the Departments.

A screenshot of a computer

Description automatically generated

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

A screenshot of a computer

Description automatically generated

* 1. Display all the distinct exam types from the EXAMS table.

**Database Foundations  
6-7: Restricting Data Using WHERE Statement  
Practices**

Tasks  
1. Display the course details for the Spring Session.  
2. Display the details of the students who have scored more than 95.  
3. Display the details of the students who have scored between 65 and 70.  
4. Display the students who registered after 01-Jun-2012.  
5. Display the course details for departments 10 and 30.  
6. Display the details of students whose first name begins with the letter "J".  
7. Display the details of students who have opted for courses 190 or 193.  
8. Display the course details offered by Department 30 for the Fall Session (Session ID 200)  
9. Display the course details of courses not being offered in the summer and fall session (Session ID 200 and 300).  
10. Display the course details for department 20.

A screenshot of a computer

Description automatically generated

* 1. Display the course details for the Spring Session
  2. Display the details of the students who have scored more than 95.A screenshot of a computer

     Description automatically generated

A screenshot of a computer

Description automatically generated

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

A screenshot of a computer

Description automatically generated

* 1. Display the students who registered after 01-Jun-2012. (in my table all students were registered in 2012).

A screenshot of a computer

Description automatically generated

* 1. Display the course details for departments 10 and 30

A screenshot of a computer

Description automatically generated

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

A screenshot of a computer

Description automatically generated

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

A screenshot of a computer

Description automatically generated

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

A screenshot of a computer

Description automatically generated

* 1. Display the course details of courses not being offered in the fall session (Session ID 200).

A screenshot of a computer

Description automatically generated

* 1. Display the course details for department 20.

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

Exercise 1: Sorting Data Using ORDER BY

1. Display all fields for each of the records in ascending order

A screenshot of a computer program

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

1. A screenshot of a computer

   Description automatically generated Display the percentage of days students have taken days off and sort the records based on the percentage calculated

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

A screenshot of a computer program

Description automatically generated

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

A screenshot of a computer

Description automatically generated

**Database Foundations  
6-9: Joining Tables Using JOIN  
Practices**

Exercise 1: Using JOINS in SQL Queries

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

A screenshot of a computer

Description automatically generated

* 1. Display the courses offered in the Fall session.

A screenshot of a computer

Description automatically generated

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

A screenshot of a computer

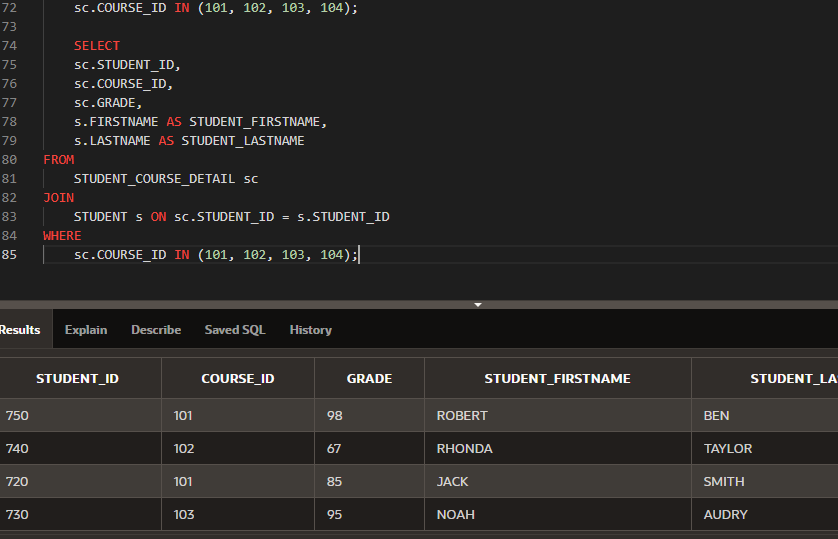
Description automatically generated

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

A screenshot of a computer program

Description automatically generated

* 1. 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



Note: Since there were no entries in the STUDENT\_COURSE\_DETAIL table for COURSE\_ID values between 190 and 192, I explored the available courses and their enrollments. I found that the existing courses included IDs 101, 102, 103, and 104, which had corresponding student enrollments. Therefore, I adjusted the SQL query to retrieve the exam grades of students enrolled in these available courses, ensuring I focused on existing data rather than non-existent course IDs. This allowed me to successfully display the relevant student grades and their associated course details.

* 1. Retrieve the rows from the EXAM\_RESULT table even if there are no matching records in the COURSE table

A screenshot of a computer

Description automatically generated

* 1. What output would be generated when the given statement is executed?  
     SELECT \* FROM AD\_EXAMS  
     CROSS JOIN AD\_EXAM\_TYPES;
* When you run the query SELECT \* FROM AD\_EXAMS CROSS JOIN AD\_EXAM\_TYPES; it will:
  1. **Combine All Rows**: Every row in the AD\_EXAMS table will be paired with every row in the AD\_EXAM\_TYPES table.
  2. If AD\_EXAMS has 3 rows and AD\_EXAM\_TYPES has 2 rows, you'll get 6 rows in total.
  3. **All Columns**: The result will include all the columns from both tables.
  4. it gives you every possible combination of rows from the two tables