**Programming 12&13**

**Database Programming with SQL  
12-1: INSERT Statements  
Practice Activities**

Vocabulary  
Identify the vocabulary word for each definition below

|  |  |
| --- | --- |
| **End User** | Someone doing “real work” with the computer, using it as a means rather than an end |
| **Transaction** | Consists of a collection of DML statements that form a logical unit of work |
| **Explicit** | Fully and clearly expressed, leaving nothing implied |
| **INSERT** | Adds a new row to a table |

Try It / Solve It  
Students should execute DESC table name before doing INSERT to view the data types for each  
column. VARCHAR2 data-type entries need single quotation marks in the VALUES statement.

1. Give two examples of why it is important to be able to alter the data in a database.

 **Example 1**: To update customer contact information. For instance, if a customer changes their phone number or email, the database needs to be updated to reflect these changes so communication can remain effective.

 **Example 2**: To maintain accurate inventory levels. In a retail system, each time an item is sold, returned, or restocked, the inventory data must be adjusted to reflect the correct quantity on hand

1. DJs on Demand just purchased four new CDs. Use an explicit INSERT statement to add each CD to the copy\_d\_cds table. After completing the entries, execute a SELECT \* statement to verify your work.

INSERT INTO copy\_d\_cds (CD\_Number, Title, Producer, Year)

VALUES (97, 'Celebrate the Day', 'R & B Inc.', 2003);

INSERT INTO copy\_d\_cds (CD\_Number, Title, Producer, Year)

VALUES (98, 'Holiday Tunes for All Ages', 'Tunes are Us', 2004);

INSERT INTO copy\_d\_cds (CD\_Number, Title, Producer, Year)

VALUES (99, 'Party Music', 'Old Town Records', 2004);

INSERT INTO copy\_d\_cds (CD\_Number, Title, Producer, Year)

VALUES (100, 'Best of Rock and Roll', 'Old Town Records', 2004);

SELECT \* FROM copy\_d\_cds;

1. DJs on Demand has two new events coming up. One event is a fall football party and the other event is a sixties theme party. The DJs on Demand clients requested the songs shown in the table for their events. Add these songs to the copy\_d\_songs table using an implicit INSERT statement.

INSERT INTO copy\_d\_songs

VALUES (52, 'Surfing Summer', NULL, 12);

INSERT INTO copy\_d\_songs

VALUES (53, 'Victory Victory', '5 min', 12);

1. Add the two new clients to the copy\_d\_clients table. Use either an implicit or an explicit INSERT.

INSERT INTO copy\_d\_clients (Client\_Number, First\_Name, Last\_Name, Phone, Email)

VALUES (6655, 'Ayako', 'Dahish', 3608859030, 'dahisha@harbor.net');

INSERT INTO copy\_d\_clients (Client\_Number, First\_Name, Last\_Name, Phone, Email)

VALUES (6689, 'Nick', 'Neuville', 9048953049, 'nnicky@charter.net');

1. Add the new client’s events to the copy\_d\_events table. The cost of each event has not been determined at this date.

INSERT INTO copy\_d\_events (ID, Name, Event\_Date, Description, Venue\_ID, Package\_Code, Theme\_Code, Client\_Number)

VALUES (110, 'Ayako Anniversary', TO\_DATE('07-JUL-2004', 'DD-MON-YYYY'), 'Party for 50, sixties dress, decorations', 245, 79, 240, 6655);

INSERT INTO copy\_d\_events (ID, Name, Event\_Date, Description, Venue\_ID, Package\_Code, Theme\_Code, Client\_Number)

VALUES (115, 'Neuville Sports Banquet', TO\_DATE('09-SEP-2004', 'DD-MON-YYYY'), 'Barbecue at residence, college alumni, 100 people', 315, 87, 340, 6689);

1. Create a table called rep\_email using the following statement:  
   CREATE TABLE rep\_email (  
   id NUMBER(3) CONSTRAINT rel\_id\_pk PRIMARY KEY,  
   first\_name VARCHAR2(10),  
   last\_name VARCHAR2(10),  
   email\_address VARCHAR2(10))

Populate this table by running a query on the employees table that includes only those employees who are REP’s.

CREATE TABLE rep\_email (

id NUMBER(3) CONSTRAINT rel\_id\_pk PRIMARY KEY,

first\_name VARCHAR2(10),

last\_name VARCHAR2(10),

email\_address VARCHAR2(10)

);

INSERT INTO rep\_email (id, first\_name, last\_name, email\_address)

SELECT employee\_id, first\_name, last\_name, email

FROM employees

WHERE job\_title = 'REP';

**Database Programming with SQL  
12-2: Updating Column Values and Deleting Rows  
Practice Activities**

Vocabulary  
Identify the vocabulary word for each definition below

Vocabulary  
Identify the vocabulary word for each definition below.

|  |  |
| --- | --- |
| **UPDATE** | Modifies existing rows in a table |
| **Subquery** | retrieves information from one table & uses the information to update another table |
| **Constraint** | Ensures that the data adheres to a predefined set of rules |
| **ON DELETE CASCADE** | deletes information on a linked table based on what was deleted on the other table |
| **DELETE** | Removes existing rows from a table |

Try It / Solve It

NOTE: Copy tables in this section do not exist  
If any change is not possible, give an explanation as to why it is not possible.

1. Monique Tuttle, the manager of Global Fast Foods, sent a memo requesting an immediate change in prices. The price for a strawberry shake will be raised from $3.59 to $3.75, and the price for fries will increase to $1.20. Make these changes to the copy\_f\_food\_items table.

UPDATE copy\_f\_food\_items

SET price = 3.75

WHERE item\_name = 'Strawberry Shake';

UPDATE copy\_f\_food\_items

SET price = 1.20

WHERE item\_name = 'Fries';

1. Bob Miller and Sue Doe have been outstanding employees at Global Fast Foods. Management has decided to reward them by increasing their overtime pay. Bob Miller will receive an additional $0.75 per hour and Sue Doe will receive an additional $0.85 per hour. Update the copy\_f\_staffs table to show these new values. (Note: Bob Miller currently doesn’t get overtime pay. What function do you need to use to convert a null value to 0?)

UPDATE copy\_f\_staffs

SET overtime\_pay = NVL(overtime\_pay, 0) + 0.75

WHERE first\_name = 'Bob' AND last\_name = 'Miller';

UPDATE copy\_f\_staffs

SET overtime\_pay = NVL(overtime\_pay, 0) + 0.85

WHERE first\_name = 'Sue' AND last\_name = 'Doe';

1. Add the orders shown to the Global Fast Foods copy\_f\_orders table:

INSERT INTO copy\_f\_orders (ORDER\_NUMBER, ORDER\_DATE, ORDER\_TOTAL, CUST\_ID, STAFF\_ID)

VALUES (5680, TO\_DATE('12-JUN-2004', 'DD-MON-YYYY'), 159.78, 145, 9);

INSERT INTO copy\_f\_orders (ORDER\_NUMBER, ORDER\_DATE, ORDER\_TOTAL, CUST\_ID, STAFF\_ID)

VALUES (5691, TO\_DATE('23-SEP-2004', 'DD-MON-YYYY'), 145.98, 225, 12);

INSERT INTO copy\_f\_orders (ORDER\_NUMBER, ORDER\_DATE, ORDER\_TOTAL, CUST\_ID, STAFF\_ID)

VALUES (5701, TO\_DATE('04-JUL-2004', 'DD-MON-YYYY'), 229.31, 230, 12);

1. Add the new customers shown below to the copy\_f\_customers table. You may already have added Katie Hernandez. Will you be able to add all these records successfully?

INSERT INTO copy\_f\_customers (ID, FIRST\_NAME, LAST\_NAME, ADDRESS, CITY, STATE, ZIP, PHONE\_NUMBER)

VALUES (145, 'Katie', 'Hernandez', '92 Chico Way', 'Los Angeles', 'CA', 98008, 8586667641);

INSERT INTO copy\_f\_customers (ID, FIRST\_NAME, LAST\_NAME, ADDRESS, CITY, STATE, ZIP, PHONE\_NUMBER)

VALUES (225, 'Daniel', 'Spode', '1923 Silverado', 'Denver', 'CO', 80219, 7193343523);

INSERT INTO copy\_f\_customers (ID, FIRST\_NAME, LAST\_NAME, ADDRESS, CITY, STATE, ZIP, PHONE\_NUMBER)

VALUES (230, 'Adam', 'Zurn', '5 Admiral Way', 'Seattle', 'WA', 4258879009);

1. Sue Doe has been an outstanding Global Foods staff member and has been given a salary raise. She will now be paid the same as Bob Miller. Update her record in copy\_f\_staffs.

UPDATE copy\_f\_staffs

SET salary = (SELECT salary FROM copy\_f\_staffs WHERE first\_name = 'Bob' AND last\_name = 'Miller')

WHERE first\_name = 'Sue' AND last\_name = 'Doe';

1. Global Fast Foods is expanding their staff. The manager, Monique Tuttle, has hired Kai Kim. Not all information is available at this time, but add the information shown here.

INSERT INTO copy\_f\_staffs (ID, FIRST\_NAME, LAST\_NAME, BIRTHDATE, SALARY, STAFF\_TYPE)

VALUES (25, 'Kai', 'Kim', TO\_DATE('03-NOV-1988', 'DD-MON-YYYY'), 6.75, 'Order Taker');

1. Now that all the information is available for Kai Kim, update his Global Fast Foods record to include the following: Kai will have the same manager as Sue Doe. He does not qualify for overtime. Leave the values for training, manager budget, and manager target as null

UPDATE copy\_f\_staffs

SET manager\_id = (SELECT manager\_id FROM copy\_f\_staffs WHERE first\_name = 'Sue' AND last\_name = 'Doe'),

overtime\_pay = NULL

WHERE first\_name = 'Kai' AND last\_name = 'Kim';

1. Execute the following SQL statement. Record your results.  
   DELETE from departments  
   WHERE department\_id = 60;

DELETE FROM departments

WHERE department\_id = 60;

1. Kim Kai has decided to go back to college and does not have the time to work and go to school. Delete him from the Global Fast Foods staff. Verify that the change was made.

DELETE FROM copy\_f\_staffs

WHERE first\_name = 'Kai' AND last\_name = 'Kim';

SELECT \* FROM copy\_f\_staffs WHERE first\_name = 'Kai' AND last\_name = 'Kim';

1. Create a copy of the employees table and call it lesson7\_emp;  
   Once this table exists, write a correlated delete statement that will delete any employees from the lesson7\_employees table that also exist in the job\_history table.

CREATE TABLE lesson7\_emp AS

SELECT \* FROM employees;

DELETE FROM lesson7\_emp

WHERE EXISTS (

SELECT 1

FROM job\_history

WHERE job\_history.employee\_id = lesson7\_emp.employee\_id

);

**Database Programming with SQL  
12-3: DEFAULT Values, MERGE, and Multi-Table Inserts  
Practice Activities**

Try It / Solve It

1. When would you want a DEFAULT value?

A DEFAULT value is useful when you want to automatically populate a column with a specific value if no value is provided during an INSERT operation. This is helpful for fields like created\_date, which might default to the current date (SYSDATE), or for a status field that might default to "active." Using DEFAULT values can simplify data entry and ensure consistent initial values across records.

1. Currently, the Global Foods F\_PROMOTIONAL\_MENUS table START\_DATE column does not have SYSDATE set as DEFAULT. Your manager has decided she would like to be able to set the starting date of promotions to the current day for some entries. This will require three steps:
2. In your schema, Make a copy of the Global Foods F\_PROMOTIONAL\_MENUS table using the  
   following SQL statement:  
   CREATE TABLE copy\_f\_promotional\_menus  
   AS (SELECT \* FROM f\_promotional\_menus)

CREATE TABLE copy\_f\_promotional\_menus

AS (SELECT \* FROM f\_promotional\_menus);

1. Alter the current START\_DATE column attributes using:  
   ALTER TABLE copy\_f\_promotional\_menus  
   MODIFY(start\_date DATE DEFAULT SYSDATE)

ALTER TABLE copy\_f\_promotional\_menus

MODIFY (start\_date DATE DEFAULT SYSDATE);

1. INSERT the new information and check to verify the results.  
   INSERT a new row into the copy\_f\_promotional\_menus table for the manager’s new  
   promotion. The promotion code is 120. The name of the promotion is ‘New Customer.’ Enter DEFAULT for the start date and '01-Jun-2005' for the ending date. The giveaway is a 10% discount coupon. What was the correct syntax used?

INSERT INTO copy\_f\_promotional\_menus (promotion\_code, name, start\_date, end\_date, giveaway)

VALUES (120, 'New Customer', DEFAULT, TO\_DATE('01-Jun-2005', 'DD-Mon-YYYY'), '10% discount coupon');

1. Allison Plumb, the event planning manager for DJs on Demand, has just given you the following list of CDs she acquired from a company going out of business. She wants a new updated list of CDs in inventory in an hour, but she doesn’t want the original D\_CDS table changed. Prepare an updated inventory list just for her.
2. Assign new cd\_numbers to each new CD acquired.

INSERT INTO manager\_copy\_d\_cds (CD\_Number, Title, Producer, Year)

VALUES (101, 'Jazz Nights', 'Blue Note Records', 1999);

INSERT INTO manager\_copy\_d\_cds (CD\_Number, Title, Producer, Year)

VALUES (102, 'Chill Vibes', 'Sunset Sounds', 2003);

INSERT INTO manager\_copy\_d\_cds (CD\_Number, Title, Producer, Year)

VALUES (103, 'Rock Legends', 'Old Town Records', 2000);

INSERT INTO manager\_copy\_d\_cds (CD\_Number, Title, Producer, Year)

VALUES (104, 'Classical Essentials', 'Harmony Studio', 2001);

1. Create a copy of the D\_CDS table called manager\_copy\_d\_cds. What was the correct syntax used?

CREATE TABLE manager\_copy\_d\_cds

AS (SELECT \* FROM d\_cds);

1. INSERT into the manager\_copy\_d\_cds table each new CD title using an INSERT statement. Make up one example or use this data:  
   20, 'Hello World Here I Am', 'Middle Earth Records', '1998'  
   What was the correct syntax used?

INSERT INTO manager\_copy\_d\_cds (CD\_Number, Title, Producer, Year)

VALUES (20, 'Hello World Here I Am', 'Middle Earth Records', 1998);

1. Use a merge statement to add to the manager\_copy\_d\_cds table, the CDs from the original table. If there is a match, update the title and year. If not, insert the data from the original table. What was the correct syntax used?

MERGE INTO manager\_copy\_d\_cds target

USING d\_cds source

ON (target.cd\_number = source.cd\_number)

WHEN MATCHED THEN

UPDATE SET target.title = source.title, target.year = source.year

WHEN NOT MATCHED THEN

INSERT (cd\_number, title, producer, year)

VALUES (source.cd\_number, source.title, source.producer, source.year);

1. Run the following 3 statements to create 3 new tables for use in a Multi-table insert statement. All 3 tables should be empty on creation, hence the WHERE 1=2 condition in the WHERE clause.

CREATE TABLE sal\_history (employee\_id, hire\_date, salary)  
AS SELECT employee\_id, hire\_date, salary  
FROM employees  
WHERE 1=2;

CREATE TABLE mgr\_history (employee\_id, manager\_id, salary)  
AS SELECT employee\_id, manager\_id, salary  
FROM employees  
WHERE 1=2;

CREATE TABLE special\_sal (employee\_id, salary)  
AS SELECT employee\_id, salary  
FROM employees  
WHERE 1=2;  
Once the tables exist in your account, write a Multi-Table insert statement to first select the employee\_id, hire\_date, salary, and manager\_id of all employees. If the salary is more than 20000 insert the employee\_id and salary into the special\_sal table. Insert the details of employee\_id, hire\_date, and salary into the sal\_history table. Insert the employee\_id,  
manager\_id, and salary into the mgr\_history table.

You should get a message back saying 39 rows were inserted. Verify you get this message and verify you have the following number of rows in each table:

Sal\_history: 19 rows  
Mgr\_history: 19 rows  
Special\_sal: 1

INSERT ALL

WHEN salary > 20000 THEN

INTO special\_sal (employee\_id, salary) VALUES (employee\_id, salary)

INTO sal\_history (employee\_id, hire\_date, salary) VALUES (employee\_id, hire\_date, salary)

INTO mgr\_history (employee\_id, manager\_id, salary) VALUES (employee\_id, manager\_id, salary)

SELECT employee\_id, hire\_date, salary, manager\_id

FROM employees;

SELECT COUNT(\*) FROM sal\_history; -- Should return 19 rows

SELECT COUNT(\*) FROM mgr\_history; -- Should return 19 rows

SELECT COUNT(\*) FROM special\_sal; -- Should return 1 row

**Database Programming with SQL  
13-1: Creating Tables  
Practice Activities**

Vocabulary  
Identify the vocabulary word for each definition below

|  |  |
| --- | --- |
| **Data Dictionary** | Created and maintained by the Oracle Server and contains information about the database |
| **Schema** | A collection of objects that are the logical structures that directly refer to the data in the database |
| **Default Value** | Specifies a preset value if a value is omitted in the INSERT statement |
| **Table** | Stores data; basic unit of storage composed of rows and columns |
| **CREATE TABLE** | Command use to make a new table |

Try It / Solve It

1. Complete the GRADUATE CANDIDATE table instance chart. Credits is a foreign-key column referencing the requirements table

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Column Name | Student\_id | Last\_name | First\_name | credits | Graduation\_date |
| Key type | Primary Key |  |  | Foreign Key |  |
| Nulls/Unique | Not Null |  |  |  |  |
| FK Column |  |  |  | Yes |  |
| Datatype | NUMBER | VARCHAR2 | VARCHAR2 | NUMBER | DATE |
| Length | 6 |  |  | 3 |  |

1. Write the syntax to create the grad\_candidates table

CREATE TABLE grad\_candidates (

student\_id NUMBER(6) PRIMARY KEY,

last\_name VARCHAR2(30),

first\_name VARCHAR2(30),

credits NUMBER REFERENCES requirements(credits),

graduation\_date DATE

);

1. Confirm creation of the table using DESCRIBE

DESCRIBE grad\_candidates;

1. Create a new table using a subquery. Name the new table your last name -- e.g., smith\_table. Using a subquery, copy grad\_candidates into smith\_table.

CREATE TABLE smith\_table AS

SELECT \* FROM grad\_candidates;

1. Insert your personal data into the table created in question 4.

INSERT INTO smith\_table (student\_id, last\_name, first\_name, credits, graduation\_date)

VALUES (123456, 'Sharma', 'Shveta', 30, TO\_DATE('2024-11-12', 'YYYY-MM-DD'));

1. Query the data dictionary for each of the following:  
   • USER\_TABLES  
   • USER\_OBJECTS  
   • USER\_CATALOG or USER\_CAT  
   In separate sentences, summarize what each query will return.

SELECT \* FROM USER\_TABLES;

Returns information about the tables owned by the current user, including table names, statuses, and storage information.

SELECT \* FROM USER\_OBJECTS;

Provides details on all objects owned by the user, such as tables, indexes, views, sequences, and constraints. It includes each object's name, type, creation date, and status.

SELECT \* FROM USER\_CATALOG;

Displays a catalog of all objects (tables, views, indexes, synonyms) owned by the current user. It's a consolidated view that summarizes the user’s schema objects.

**Database Programming with SQL  
13-2: Using Data Types  
Practice Activities**

Vocabulary  
Identify the vocabulary word for each definition below.

|  |  |
| --- | --- |
| **INTERVAL YEAR TO MONTH** | Allows time to be stored as an interval of years and months |
| **TIMESTAMP WITH LOCAL TIME ZONE** | When a column is selected in a SQL statement the time is automatically converted to the user’s timezone |
| **BLOB** | Binary large object data up to 4 gigabytes |
| **TIMESTAMP WITH TIME ZONE** | Stores a time zone value as a displacement from Universal Coordinated Time or UCT |
| **INTERVAL DAY TO SECOND** | Allows time to be stored as an interval of days to hours, minutes, and seconds |
| **CLOB** | Character data up to 4 gigabytes |
| **TIMESTAMP** | Allows the time to be stored as a date with fractional seconds |

Try It / Solve It

1. Create tables using each of the listed time-zone data types, use your time-zone and one other in your examples. Answers will vary.

a. TIMESTAMP WITH LOCAL TIME ZONE

CREATE TABLE timezone\_local\_example (

event\_id NUMBER PRIMARY KEY,

event\_time TIMESTAMP WITH LOCAL TIME ZONE

);

INSERT INTO timezone\_local\_example (event\_id, event\_time)

VALUES (1, TIMESTAMP '2024-11-12 14:30:00 US/Eastern');

INSERT INTO timezone\_local\_example (event\_id, event\_time)

VALUES (2, TIMESTAMP '2024-11-12 14:30:00 Europe/London');

b. INTERVAL YEAR TO MONTH

CREATE TABLE interval\_year\_month\_example (

interval\_id NUMBER PRIMARY KEY,

duration INTERVAL YEAR TO MONTH

);

INSERT INTO interval\_year\_month\_example (interval\_id, duration)

VALUES (1, INTERVAL '2-6' YEAR TO MONTH);

c. INTERVAL DAY TO SECOND

CREATE TABLE interval\_day\_second\_example (

interval\_id NUMBER PRIMARY KEY,

duration INTERVAL DAY TO SECOND

);

INSERT INTO interval\_day\_second\_example (interval\_id, duration)

VALUES (1, INTERVAL '15 12:30:45' DAY TO SECOND);

2. Execute a SELECT \* from each table to verify your input.

SELECT \* FROM timezone\_local\_example;

SELECT \* FROM interval\_year\_month\_example;

SELECT \* FROM interval\_day\_second\_example;

3. Give 3 examples of organizations and personal situations where it is important to know to which time zone a date-time value refers.

**Airlines and Travel Agencies**  
Airlines need precise time zone information to manage flight schedules and ensure coordination of arrivals and departures across multiple locations.

**International Financial Institutions**  
Banks and stock exchanges operate across time zones and need accurate date-time tracking to handle transactions, investments, and trading sessions.

**Telemedicine and Remote Healthcare**  
Telemedicine providers need to schedule and manage appointments across different time zones to coordinate between healthcare professionals and patients accurately.

**Database Programming with SQL  
13-3: Modifying a Table  
Practice Activities**

Try It / Solve It

Before beginning the practice exercises, execute a DESCRIBE for each of the following tables:  
o\_employees, o\_departments and o\_jobs. These tables will be used in the exercises. If they do not exist in your account, create them as follows:

**1. Create the three o\_tables – jobs, employees, and departments – using the syntax:**

**CREATE TABLE o\_jobs AS (SELECT \* FROM jobs);  
CREATE TABLE o\_employees AS (SELECT \* FROM employees);  
CREATE TABLE o\_departments AS (SELECT \* FROM departments);**

**2. Add the Human Resources job to the jobs table:**

**INSERT INTO o\_jobs (job\_id, job\_title, min\_salary, max\_salary)  
VALUES('HR\_MAN', 'Human Resources Manager', 4500, 5500);**

**3. Add the three new employees to the employees table:**

**INSERT INTO o\_employees (employee\_id, first\_name, last\_name, email, hire\_date,  
job\_id)  
VALUES(210, 'Ramon', 'Sanchez', 'RSANCHEZ', SYSDATE, 'HR\_MAN');**

**4. Add Human Resources to the departments table:**

**INSERT INTO o\_departments(department\_id, department\_name)  
VALUES (210,'Human Resources');**

You will need to know which columns do not allow null values.  
1. Why is it important to be able to modify a table?

Modifying a table allows you to adapt the structure to changing data requirements. For example, you may need to add columns to store new types of information, remove unused columns, adjust data types for efficiency, or rename columns for clarity. These modifications are essential to keep the database relevant and optimized for current business needs.

2. CREATE a table called Artists.

a. Add the following to the table:  
• artist ID  
• first name  
• last name  
• band name  
• email  
• hourly rate

CREATE TABLE Artists (

artist\_id NUMBER PRIMARY KEY,

first\_name VARCHAR2(30),

last\_name VARCHAR2(30),

band\_name VARCHAR2(50),

email VARCHAR2(50),

hourly\_rate NUMBER(5, 2)

);

b. INSERT one artist from the d\_songs table.

INSERT INTO Artists (artist\_id, first\_name, last\_name, band\_name, email, hourly\_rate)

SELECT 1, 'John', 'Doe', 'Rock Band', 'johndoe@example.com', 50

FROM d\_songs

WHERE rownum = 1;

c. INSERT one artist of your own choosing.

INSERT INTO Artists (artist\_id, first\_name, last\_name, band\_name, email, hourly\_rate)

VALUES (2, 'Jane', 'Smith', 'Jazz Ensemble', 'janesmith@example.com', 60);

d. Give an example how each of the following may be used on the table that you have created:  
1) ALTER TABLE

ALTER TABLE Artists ADD genre VARCHAR2(30);

2) DROP TABLE

DROP TABLE Artists;

3) RENAME TABLE

RENAME Artists TO Musicians;

4) TRUNCATE

TRUNCATE TABLE Artists;

5) COMMENT ON TABLE

COMMENT ON TABLE Artists IS 'This table stores artist information for our music app';

3. In your o\_employees table, enter a new column called “Termination.” The datatype for the newcolumn should be VARCHAR2. Set the DEFAULT for this column as SYSDATE to appear as character data in the format: February 20th, 2003.

ALTER TABLE o\_employees

ADD Termination VARCHAR2(30) DEFAULT TO\_CHAR(SYSDATE, 'Month DDth, YYYY');

4. Create a new column in the o\_employees table called start\_date. Use the TIMESTAMP WITH  
LOCAL TIME ZONE as the datatype.

ALTER TABLE o\_employees

ADD start\_date TIMESTAMP WITH LOCAL TIME ZONE;

5. Truncate the o\_jobs table. Then do a SELECT \* statement. Are the columns still there? Is the data still there?

TRUNCATE TABLE o\_jobs;

SELECT \* FROM o\_jobs;

**Results**: The columns remain, but all data rows are removed.

6. What is the distinction between TRUNCATE, DELETE, and DROP for tables?

**TRUNCATE**: Removes all rows from a table without logging individual deletions. It’s fast but cannot be rolled back.

**DELETE**: Removes specified rows with individual logging. It can be rolled back if within a transaction.

**DROP**: Deletes the table entirely, removing both data and table structure permanently.

7. List the changes that can and cannot be made to a column.

**Allowed**:

* Add new columns.
* Modify data type or size if there’s no data loss.
* Drop columns.

**Not Allowed**:

* Change the primary key to NULL.
* Reduce data type size if data loss would occur.

8. Add the following comment to the o\_jobs table: "New job description added"  
View the data dictionary to view your comments.

COMMENT ON TABLE o\_jobs IS 'New job description added';

SELECT comments FROM user\_tab\_comments WHERE table\_name = 'O\_JOBS';

9. Rename the o\_jobs table to o\_job\_description.

RENAME o\_jobs TO o\_job\_description;

10. F\_staffs table exercises:

a. Create a copy of the f\_staffs table called copy\_f\_staffs and use this copy table for the  
remaining labs in this lesson.

CREATE TABLE copy\_f\_staffs AS SELECT \* FROM f\_staffs;

b. Describe the new table to make sure it exists.

DESCRIBE copy\_f\_staffs;

c. Drop the table.

DROP TABLE copy\_f\_staffs;

d. Try to select from the table.

SELECT \* FROM copy\_f\_staffs;

e. Investigate your recyclebin to see where the table went.

SHOW RECYCLEBIN;

f. Try to select from the dropped table by using the value stored in the OBJECT\_NAME column.  
You will need to copy and paste the name as it is exactly, and enclose the new name in “ “  
(double quotes). So if the dropped name returned to you is  
BIN$Q+x1nJdcUnngQESYELVIdQ==$0, you need to write a query that refers to  
“BIN$Q+x1nJdcUnngQESYELVIdQ==$0”.

SELECT \* FROM "BIN$Q+x1nJdcUnngQESYELVIdQ==$0";

g. Undrop the table.

FLASHBACK TABLE copy\_f\_staffs TO BEFORE DROP;

h. Describe the table.

DESCRIBE copy\_f\_staffs;

11. Still working with the copy\_f\_staffs table, perform an update on the table.

a. Issue a select statement to see all rows and all columns from the copy\_f\_staffs table;

SELECT \* FROM copy\_f\_staffs;

b. Change the salary for Sue Doe to 12 and commit the change.

UPDATE copy\_f\_staffs SET salary = 12 WHERE first\_name = 'Sue' AND last\_name = 'Doe';

COMMIT;

c. Issue a select statement to see all rows and all columns from the copy\_f\_staffs table;

SELECT \* FROM copy\_f\_staffs;

d. For Sue Doe, update the salary to 2 and commit the change.

UPDATE copy\_f\_staffs SET salary = 2 WHERE first\_name = 'Sue' AND last\_name = 'Doe';

COMMIT;

e. Issue a select statement to see all rows and all columns from the copy\_f\_staffs table;

SELECT \* FROM copy\_f\_staffs;

f. Now, issue a FLASHBACK QUERY statement against the copy\_f\_staffs table, so you can see  
all the changes made.

SELECT \* FROM copy\_f\_staffs AS OF TIMESTAMP (SYSTIMESTAMP - INTERVAL '1' MINUTE)

WHERE first\_name = 'Sue' AND last\_name = 'Doe';

g. Investigate the result of f), and find the original salary and update the copy\_f\_staffs table salary column for Sue Doe back to her original salary

UPDATE copy\_f\_staffs SET salary = 10 WHERE first\_name = 'Sue' AND last\_name = 'Doe';

COMMIT;