EXERCISE 11

Find the Solution for the following:

1. Create a view called EMPLOYEE_VU based on the employee numbers, employee names and department numbers from the EMPLOYEES table. Change the heading for the employee name to EMPLOYEE.

```
CREATE VIEW EMPLOYEE_VU AS
SELECT employee_id AS empno, employee_name AS employee, department_id AS deptno
FROM EMPLOYEES;
```

2. Display the contents of the EMPLOYEES_VU view.

```
SELECT * FROM EMPLOYEE_VU;
```

3. Select the view name and text from the USER_VIEWS data dictionary views.

```
SELECT view_name, text FROM USER_VIEWS WHERE view_name = 'EMPLOYEE_VU';
```

4. Using your EMPLOYEES_VU view, enter a query to display all employees names and Department.

```
SELECT employee, deptno FROM EMPLOYEE_VU;
```

5. Create a view named DEPT50 that contains the employee number, employee last names and department numbers for all employees in department 50.Label the view columns EMPNO, EMPLOYEE and DEPTNO. Do not allow an employee to be reassigned to another department through the view.

```
CREATE VIEW DEPT50 AS
SELECT employee_id AS empno, last_name AS employee, department_id AS deptno
FROM EMPLOYEES
WHERE department_id = 50;
```

6. Display the structure and contents of the DEPT50 view.

```
DESCRIBE DEPT50;
SELECT * FROM DEPT50;
```

7. Attempt to reassign Matos to department 80.

Since DEPT50 view only allows employees from department 50 to be shown, you can't reassign Matos to department 80 using this view directly

8. Create a view called SALARY_VU based on the employee last names, department names, salaries, and salary grades for all employees. Use the Employees, DEPARTMENTS and

JOB_GRADE tables. Label the column Employee, Department, salary, and Grade respectively.

```
CREATE VIEW SALARY_VU AS
SELECT e.last_name AS Employee, d.department_name AS Department, e.salary, jg.grade
FROM EMPLOYEES e
JOIN DEPARTMENTS d ON e.department_id = d.department_id
JOIN JOB_GRADE jg ON e.salary BETWEEN jg.lowest_sal AND jg.highest_sal;
```

Practice Problems -I

JOIN CLAUSES

Use the Oracle database for problems 1-6.
1. Join the Oracle database locations and departments table using the location_id column. Limit the results to location 1400 only.

```
SELECT l.*, d.*
FROM locations l
JOIN departments d ON l.location_id = d.location_id
WHERE l.location_id = 1400;
```

2. Join DJs on Demand d_play_list_items, d_track_listings, and d_cds tables with the JOIN USING syntax. Include the song ID, CD number, title, and comments in the output.

```
SELECT pli.song_id, tl.cd_number, tl.title, pli.comments
FROM d_play_list_items pli
JOIN d_track_listings tl USING (song_id)
JOIN d_cds c USING (cd_number);
```

3. Display the city, department name, location ID, and department ID for departments 10, 20, and 30 for the city of Seattle.

```
SELECT l.city, d.department_name, d.location_id, d.department_id
FROM locations l
JOIN departments d ON l.location_id = d.location_id
WHERE l.city = 'Seattle' AND d.department_id IN (10, 20, 30);
```

4. Display country name, region ID, and region name for Americas.

```
SELECT c.country_name, r.region_id, r.region_name
FROM countries c
JOIN regions r ON c.region_id = r.region_id
WHERE r.region_name = 'Americas';
```

5. Write a statement joining the employees and jobs tables. Display the first and last names, hire date, job id, job title, and maximum salary. Limit the query to those employees who are in jobs that earn more than $12,000.

```
SELECT e.first_name, e.last_name, e.hire_date, e.job_id, j.job_title, j.max_salary
FROM employees e
JOIN jobs j ON e.job_id = j.job_id
WHERE j.max_salary > 12000;
```

Inner versus Outer Joins

Use the Oracle database for problems 1-7.
1. Return the first name, last name, and department name for all employees including those employees not assigned to a department.

```
SELECT e.first_name, e.last_name, d.department_name
FROM employees e
LEFT JOIN departments d ON e.department_id = d.department_id;
```

2. Return the first name, last name, and department name for all employees including those departments that do not have an employee assigned to them.

```
SELECT e.first_name, e.last_name, d.department_name
FROM employees e
RIGHT JOIN departments d ON e.department_id = d.department_id;
```

3. Return the first name, last name, and department name for all employees including those departments that do not have an employee assigned to them and those employees not assigned to a Department.

```
SELECT e.first_name, e.last_name, d.department_name
FROM employees e
FULL OUTER JOIN departments d ON e.department_id = d.department_id;
```

4. Create a query of the DJs on Demand database to return the first name, last name, event date, and description of the event the client held. Include all the clients even if they have not had an event Scheduled.

```
SELECT c.first_name, c.last_name, e.event_date, e.description
FROM d_clients c
LEFT JOIN d_events e ON c.client_id = e.client_id;
```

5. Using the Global Fast Foods database, show the shift description and shift assignment date even if there is no date assigned for each shift description.

```
SELECT s.shift_description, sa.shift_assignment_date
FROM shifts s
LEFT JOIN shift_assignments sa ON s.shift_id = sa.shift_id;
```

Self Joins and Hierarchical Queries

For each problem, use the Oracle database.
1. Display the employee's last name and employee number along with the manager's last name and manager number. Label the columns: Employee, Emp#, Manager, and Mgr#, respectively.

```
SELECT e.last_name AS Employee, e.employee_id AS Emp#,
m.last_name AS Manager, m.employee_id AS Mgr#
FROM employees e
JOIN employees m ON e.manager_id = m.employee_id;
```

2. Modify question 1 to display all employees and their managers, even if the employee does not have a manager. Order the list alphabetically by the last name of the employee.

```
SELECT e.last_name AS Employee, e.employee_id AS Emp#,
 m.last_name AS Manager, m.employee_id AS Mgr#
FROM employees e
LEFT JOIN employees m ON e.manager_id = m.employee_id
ORDER BY e.last_name;
```

3. Display the names and hire dates for all employees who were hired before their managers, along with their managers' names and hire dates. Label the columns Employee, Emp Hired, Manager, and Mgr Hired, respectively.

```
SELECT e.last_name AS Employee, e.hire_date AS Emp_Hired,
 m.last_name AS Manager, m.hire_date AS Mgr_Hired
FROM employees e
JOIN employees m ON e.manager_id = m.employee_id
WHERE e.hire_date < m.hire_date;
```

4. Write a report that shows the hierarchy for Lex De Haans department. Include last name, salary, and department id in the report.

```
SELECT last_name, salary, department_id
FROM employees
START WITH last_name = 'De Haan'
CONNECT BY PRIOR employee_id = manager_id;
```

5. What is wrong in the following statement:

```
SELECT last_name, department_id, salary
FROM employees
START WITH last_name = 'King'
CONNECT BY PRIOR manager_id = employee_id;
```

```
SELECT last_name, department_id, salary
FROM employees
START WITH last_name = 'King'
CONNECT BY PRIOR manager_id = employee_id;
```

6. Create a report that shows the organization chart for the entire employee table. Write the report so that each level will indent each employee 2 spaces. Since Oracle Application Express cannot display the spaces in front of the column, use - (minus) instead.

```
SELECT LPAD(' ', LEVEL*2) || last_name AS Employee, department_id, salary
FROM employees
START WITH manager_id IS NULL
CONNECT BY PRIOR employee_id = manager_id;
```

7. Re-write the report from 6 to exclude De Haan and all the people working for him.

```
SELECT LPAD(' ', LEVEL*2) || last_name AS Employee, department_id, salary
FROM employees
START WITH manager_id IS NULL AND last_name != 'De Haan'
CONNECT BY PRIOR employee_id = manager_id
AND PRIOR last_name != 'De Haan';
```

Oracle Equijoin and Cartesian Product

1. Create a Cartesian product that displays the columns in the d_play_list_items and the d_track_listings in the DJs on Demand database.

```
SELECT *
FROM d_play_list_items, d_track_listings;
```

2. Correct the Cartesian product produced in question 1 by creating an equijoin using a common Column.

```
SELECT pli.*, tl.*
FROM d_play_list_items pli
JOIN d_track_listings tl ON pli.song_id = tl.song_id;
```

3. Write a query to display the title, type, description, and artist from the DJs on Demand database.

```
SELECT t.title, t.type, t.description, a.artist_name
FROM d_track_listings t
JOIN d_artists a ON t.artist_id = a.artist_id;
```

4. Rewrite the query in question 3 to select only those titles with an ID of 47 or 48.

```
SELECT t.title, t.type, t.description, a.artist_name
FROM d_track_listings t
JOIN d_artists a ON t.artist_id = a.artist_id
WHERE t.track_id IN (47, 48);
```

5. Write a query that extracts information from three tables in the DJs on Demand database, the d_clients table, the d_events table, and the d_job_assignments table.

```
SELECT c.client_name, e.event_date, ja.job_description
FROM d_clients c
JOIN d_events e ON c.client_id = e.client_id
JOIN d_job_assignments ja ON e.event_id = ja.event_id;
```

Group Functions

1. Define and give an example of the seven group functions: AVG, COUNT, MAX, MIN, STDDEV, SUM,and VARIANCE.

```
SELECT AVG(salary) FROM employees;
SELECT COUNT(*) FROM employees;
SELECT MAX(salary) FROM employees;
SELECT MIN(salary) FROM employees;
SELECT STDDEV(salary) FROM employees;
SELECT SUM(salary) FROM employees;
SELECT VARIANCE(salary) FROM employees;
```

2. Create a query that will show the average cost of the DJs on Demand events. Round to two decimal places.

```
SELECT ROUND(AVG(event_cost), 2) AS avg_cost
FROM d_events;
```

3. Find the average salary for Global Fast Foods staff members whose manager ID is 19.

```
SELECT AVG(salary) AS avg_salary
FROM Global_Fast_Foods_Staff
WHERE manager_id = 19;
```

4. Find the sum of the salaries for Global Fast Foods staff members whose IDs are 12 and 9.

```
SELECT SUM(salary) AS total_salary
FROM Global_Fast_Foods_Staff
WHERE staff_id IN (12, 9);
```

Using the Oracle database, select the lowest salary, the most recent hire date, the last name of the personwho is at the top of an alphabetical list of employees, and the last name of the person who is at the bottom of an alphabetical list of employees.
Select only employees who are in departments 50 or 60

```
SELECT MIN(salary) AS lowest_salary,
    MAX(hire_date) AS most_recent_hire_date,
    MIN(last_name) AS first_alphabetically,
    MAX(last_name) AS last_alphabetically
FROM employees
WHERE department_id IN (50, 60);
```

5. Your new Internet business has had a good year financially. You have had 1,289 orders this year. Your customer order table has a column named total_sales. If you submit the following query, how many rows will be returned?
SELECT sum(total_sales) fFROM orders;

1 row will be returned.

6. You were asked to create a report of the average salaries for all employees in each division of the company. Some employees in your company are paid hourly instead of by salary. When you ran the report, it seemed as though the averages were not what you expected—they were much higher than you thought! What could have been the cause?

Including hourly-paid employees in the average calculation can skew the results.
Ensure only salaried employees are included.

7. Employees of Global Fast Foods have birth dates of July 1, 1980, March 19, 1979, and March 30, 1969. If you select MIN(birthdate), which date will be returned?

March 30, 1969.

8. Create a query that will return the average order total for all Global Fast Foods orders from January 1, 2002, to December 21, 2002.

SELECT AVG(order_total) AS avg_order_total
FROM Global_Fast_Foods_Orders
WHERE order_date BETWEEN '01-JAN-2002' AND '21-DEC-2002';

9. What was the hire date of the last Oracle employee hired?

SELECT MAX(hire_date) AS last_hire_date
FROM employees;

10. Your new Internet business has had a good year financially. You have had 1,289 orders this year. Your customer order table has a column named total_sales. If you submit the following query, how many rows will be returned?
SELECT sum(total_sales)
FROM orders;

1row will be returned.

Practice Problems -II

COUNT, DISTINCT, NVL

1. How many songs are listed in the DJs on Demand D_SONGS table?

```
SELECT COUNT(*) AS total_songs
FROM D_SONGS;
```

2. In how many different location types has DJs on Demand had venues?

```
SELECT COUNT(DISTINCT loc_type) AS different_location_types
FROM D_VENUES;
```

3. The d_track_listings table in the DJs on Demand database has a song_id column and a cd_number column. How many song IDs are in the table and how many different CD numbers are in the table?

```
SELECT COUNT(song_id) AS total_song_ids, COUNT(DISTINCT cd_number)
 AS different_cd_numbers FROM d_track_listings;
```

4. How many of the DJs on Demand customers have email addresses?

```
SELECT COUNT(email_address) AS customers_with_emails
FROM d_customers
WHERE email_address IS NOT NULL;
```

5. Some of the partners in DJs on Demand do not have authorized expense amounts (auth_expense_amt). How many partners do have this privilege?

```
SELECT COUNT(auth_expense_amt) AS partners_with_expense_privilege
FROM d_partners
WHERE auth_expense_amt IS NOT NULL;
```

6. What values will be returned when the statement below is issued?

ID type shoe_color
456 oxford brown
463 sandal tan
262 heel black
433 slipper tan

```
SELECT COUNT(shoe_color),
COUNT(DISTINCT shoe_color)
FROM shoes;
```

COUNT(shoe_color) will return 4.
COUNT(DISTINCT shoe_color) will return 3 (brown, tan, black).

7. Create a query that will convert any null values in the auth_expense_amt column on the DJs on Demand D_PARTNERS table to 100000 and find the average of the values in this column. Round the result to two decimal places.

```
SELECT ROUND(AVG(NVL(auth_expense_amt, 100000)), 2) AS avg_expense_amt
FROM d_partners;
```

8. Which of the following statements is/are TRUE about the following query?

```
SELECT AVG(NVL(selling_bonus, 0.10))
FROM bonuses;
```

a. The datatypes of the values in the NVL clause can be any datatype except date data.
b. If the selling_bonus column has a null value, 0.10 will be substituted.
c. There will be no null values in the selling_bonus column when the average is calculated.
d. This statement will cause an error. There cannot be two functions in the SELECT Statement.

b. If the selling_bonus column has a null value, 0.10 will be substituted.
c. There will be no null values in the selling_bonus column when the average is calculated.

9. Which of the following statements is/are TRUE about the following query?

```
SELECT DISTINCT colors, sizes
FROM items;
```

a. Each color will appear only once in the results set.
b. Each size will appear only once in the results set.
c. Unique combinations of color and size will appear only once in the results set.
d. Each color and size combination will appear more than once in the results set.

c. Unique combinations of color and size will appear only once in the results set.

Using GROUP BY and HAVING Clauses

1. In the SQL query shown below, which of the following are true about this query?
a. Kimberly Grant would not appear in the results set.
b. The GROUP BY clause has an error because the manager_id is not listed in the SELECT clause.
c. Only salaries greater than 16001 will be in the result set.
d. Names beginning with Ki will appear after names beginning with Ko.
e. Last names such as King and Kochhar will be returned even if they don't have salaries > 16000.

```
SELECT last_name, MAX(salary)
FROM employees
WHERE last_name LIKE 'K%' GROUP
BY manager_id, last_name HAVING
MAX(salary) >16000
ORDER BY last_name DESC ;
```

c. Only salaries greater than 16001 will be in the result set.
e. Last names such as King and Kochhar will be returned even if they don't have salaries > 16000.

2. Each of the following SQL queries has an error. Find the error and correct it. Use Oracle Application Express to verify that your corrections produce the desired results.
a.

```
SELECT manager_id
FROM employees
WHERE AVG(salary) <16000
GROUP BY manager_id;
```

Error: AVG is a group function and should be in HAVING, not WHERE.

b.

```
SELECT cd_number, COUNT(title)
FROM d_cds
WHERE cd_number < 93;
```

Error: Missing GROUP BY clause for COUNT.

c.

```
SELECT ID, MAX(ID), artist AS Artist FROM d_songs
WHERE duration IN('3 min', '6 min', '10 min')
HAVING ID < 50
GROUP by ID;
```

Error: HAVING clause should come after GROUP BY.

d.

```
SELECT loc_type, rental_fee AS Fee
FROM d_venues
WHERE id <100
GROUP BY "Fee"
ORDER BY 2;
```

Error: GROUP BY should reference loc_type, not the alias

3. Rewrite the following query to accomplish the same result:
SELECT DISTINCT MAX(song_id)
FROM d_track_listings WHERE
track IN (1, 2, 3);

SELECT MAX(song_id)
FROM d_track_listings
WHERE track IN (1, 2, 3)
GROUP BY track;

4. Indicate True or False
a. If you include a group function and any other individual columns in a SELECT clause, then each individual column must also appear in the GROUP BY clause.

TRUE

b. You can use a column alias in the GROUP BY clause.

FALSE

c. The GROUP BY clause always includes a group function.

FALSE

5. Write a query that will return both the maximum and minimum average salary grouped by department from the employees table.

SELECT department_id, MAX(AVG(salary)) AS max_avg_salary,
MIN(AVG(salary)) AS min_avg_salary
FROM employees
GROUP BY department_id;

6. Write a query that will return the average of the maximum salaries in each department for the employees table.

SELECT AVG(max_salary) AS avg_max_salary
FROM (SELECT MAX(salary) AS max_salary FROM employees
GROUP BY department_id);

Using Set Operators

1. Name the different Set operators?

UNION
UNION ALL
INTERSECT
MINUS

2. Write one query to return the employee_id, job_id, hire_date, and department_id of all employees and a second query listing employee_id, job_id, start_date, and department_id from the job_history table and combine the results as one single output. Make sure you suppress duplicates in the output.

```
SELECT employee_id, job_id, hire_date, department_id
FROM employees
UNION
SELECT employee_id, job_id, start_date AS hire_date, department_id
FROM job_history;
```

3. Amend the previous statement to not suppress duplicates and examine the output. How many extra rows did you get returned and which were they? Sort the output by employee_id to make it easier to spot.

```
SELECT employee_id, job_id, hire_date, department_id
FROM employees
UNION ALL
SELECT employee_id, job_id, start_date AS hire_date, department_id
FROM job_history
ORDER BY employee_id;
```

There is one extra row on employee 176 with a job_id of SA_REP.
4. List all employees who have not changed jobs even once. (Such employees are not found in the job_history table)

```
SELECT employee_id, job_id, hire_date, department_id
FROM employees
WHERE employee_id NOT IN (SELECT employee_id FROM job_history);
```

5. List the employees that HAVE changed their jobs at least once.

```
SELECT employee_id, job_id, hire_date, department_id
FROM employees
WHERE employee_id IN (SELECT employee_id FROM job_history);
```

6. Using the UNION operator, write a query that displays the employee_id, job_id, and salary of ALL present and past employees. If a salary is not found, then just display a 0 (zero) in its place.

```
SELECT employee_id, job_id, salary
FROM employees
UNION
SELECT employee_id, job_id, 0 AS salary
FROM job_history;
```

Practice Problems -III

Fundamentals of Subqueries

1. What is the purpose of using a subquery?

A subquery is used to return data that will be used in the main query as a condition to further restrict the data to be retrieved.

2. What is a subquery?

A subquery, also known as an inner query or nested query, is a query within another SQL query and embedded within the WHERE clause.

3. What DJs on Demand d_play_list_items song_id's have the same event_id as song_id 45?

```
SELECT song_id
FROM d_play_list_items
WHERE event_id = (
   SELECT event_id
   FROM d_play_list_items
   WHERE song_id = 45
);
```

4. Which events in the DJs on Demand database cost more than event_id = 100?

```
SELECT event_id, cost
FROM d_events
WHERE cost > (
   SELECT cost
   FROM d_events
   WHERE event_id = 100
);
```

5. Find the track number of the song that has the same CD number as "Party Music for All Occasions."

```
SELECT track_number
FROM d_track_listings
WHERE cd_number = (
   SELECT cd_number
   FROM d_cds
   WHERE title = 'Party Music for All Occasions'
);
```

6. List the DJs on Demand events whose theme code is the same as the code for “Tropical.”

```
SELECT event_id, event_name
FROM d_events
WHERE theme_code = (
   SELECT theme_code
   FROM d_themes
   WHERE theme_name = 'Tropical'
);
```

7. What are the names of the Global Fast Foods staff members whose salaries are greater than the staff member whose ID is 12?

```
SELECT first_name, last_name
FROM gff_staff
WHERE salary > (
   SELECT salary
   FROM gff_staff
   WHERE staff_id = 12
);
```

8. What are the names of the Global Fast Foods staff members whose staff types are not the same as Bob Miller’s?

```
SELECT first_name, last_name
FROM gff_staff
WHERE staff_type <> (
   SELECT staff_type
   FROM gff_staff
   WHERE first_name = 'Bob' AND last_name = 'Miller'
);
```

9. Which Oracle employees have the same department ID as the IT department?

```
SELECT first_name, last_name
FROM employees
WHERE department_id = (
   SELECT department_id
   FROM departments
   WHERE department_name = 'IT'
);
```

10. What are the department names of the Oracle departments that have the same location ID as Seattle?

```
SELECT department_name
FROM departments
WHERE location_id = (
   SELECT location_id
   FROM locations
   WHERE city = 'Seattle'
);
```

11. Which statement(s) regarding subqueries is/are true?
a. It is good programming practice to place a subquery on the right side of the comparison operator.

TRUE

b. A subquery can reference a table that is not included in the outer query's FROM clause.

TRUE

c. Single-row subqueries can return multiple values to the outer query.

FALSE

Single-Row Subqueries

1. Write a query to return all those employees who have a salary greater than that of Lorentz and are in the same department as Abel.

```
SELECT first_name, last_name, salary, department_id
FROM employees WHERE salary > (SELECT salary FROM employees
WHERE last_name = 'Lorentz') AND department_id = (SELECT department_id
FROM employees WHERE last_name = 'Abel');
```

2. Write a query to return all those employees who have the same job id as Rajs and were hired after Davies.

```
SELECT first_name, last_name, hire_date, job_id FROM employees
WHERE job_id = (SELECT job_id FROM employees WHERE last_name = 'Rajs')
AND hire_date > (SELECT hire_date FROM employees WHERE last_name = 'Davies');
```

3. What DJs on Demand events have the same theme code as event ID = 100?

```
SELECT event_id, event_name FROM d_events
WHERE theme_code = (SELECT theme_code FROM d_events WHERE event_id = 100);
```

4. What is the staff type for those Global Fast Foods jobs that have a salary less than those of any Cook staff-type jobs?

```
SELECT staff_type FROM gff_jobs WHERE salary < (SELECT MIN(salary)
FROM gff_jobs WHERE staff_type = 'Cook');
```

5. Write a query to return a list of department id's and average salaries where the department's average salary is greater than Ernst's salary.

```
SELECT department_id, AVG(salary) AS avg_salary FROM employees
GROUP BY department_id HAVING AVG(salary) > (SELECT salary
FROM employees WHERE last_name = 'Ernst');
```

6. Return the department ID and minimum salary of all employees, grouped by department ID, having a minimum salary greater than the minimum salary of those employees whose department ID is not equal to 50.

```
SELECT department_id, MIN(salary) AS min_salary FROM employees
GROUP BY department_id HAVING MIN(salary) > (SELECT MIN(salary)
FROM employees WHERE department_id <> 50);
```

Multiple-Row Subqueries

1. What will be returned by a query if it has a subquery that returns a null?

If the subquery returns a null, the outer query might return no rows if the condition depends on the subquery's result being non-null.

2. Write a query that returns jazz and pop songs. Write a multi-row subquery and use the d_songs and d_types tables. Include the id, title, duration, and the artist name.

SELECT id, title, duration, artist FROM d_songs WHERE type_id IN (SELECT type_id FROM d_types WHERE type_name IN ('Jazz', 'Pop'));

3. Find the last names of all employees whose salaries are the same as the minimum salary for any Department.

SELECT last_name FROM employees WHERE salary IN (SELECT MIN(salary) FROM employees GROUP BY department_id);

4. Which Global Fast Foods employee earns the lowest salary? Hint: You can use either a single- row or a multiple-row subquery.

SELECT first_name, last_name FROM gff_staff WHERE salary = (SELECT MIN(salary) FROM gff_staff);

5. Place the correct multiple-row comparison operators in the outer query WHERE clause of each of the following:
a. Which CDs in our d_cds collection were produced before “Carpe Diem” was produced?
WHERE year _ (SELECT year …

WHERE year < (SELECT year FROM d_cds WHERE title = 'Carpe Diem')

b. Which employees have salaries lower than any one of the programmers in the IT department?
WHERE salary _(SELECT salary …

WHERE salary < ANY (SELECT salary FROM employees WHERE job_id = 'Programmer' AND department_id = (SELECT department_id FROM departments WHERE department_name = 'IT'))

c. What CD titles were produced in the same year as “Party Music for All Occasions” or “Carpe Diem”?
WHERE year _ (SELECT year …

WHERE year IN (SELECT year FROM d_cds WHERE title IN ('Party Music for All Occasions', 'Carpe Diem'))

d. What song title has a duration longer than every type code 77 title?
WHERE duration _ _ (SELECT duration …

WHERE duration > ALL (SELECT duration FROM d_songs WHERE type_code = 77)

6. If each WHERE clause is from the outer query, which of the following are true?
a. WHERE size > ANY -- If the inner query returns sizes ranging from 8 to 12, the value 9 could be returned in the outer query.

TRUE

b. WHERE book_number IN -- If the inner query returns books numbered 102, 105, 437, and 225 then 325 could be returned in the outer query.

FALSE

c. WHERE score <= ALL -- If the inner query returns the scores 89, 98, 65, and 72, then 82 could be returned in the outer query.

TRUE

d. WHERE color NOT IN -- If the inner query returns red, green, blue, black, and then the outer query could return white.

TRUE

e. WHERE game_date = ANY -- If the inner query returns 05-Jun-1997, 10-Dec-2002, and 2-Jan- 2004,then the outer query could return 10-Sep-2002.

FALSE

7. The goal of the following query is to display the minimum salary for each department whose minimum salary is less than the lowest salary of the employees in department 50. However, the subquery does not execute because it has five errors. Find them, correct them, and run the query.
SELECT department_id
FROM employees WHERE
MIN(salary) HAVING
MIN(salary) > GROUP BY
department_id SELECT
MIN(salary)
WHERE department_id < 50;

```
SELECT department_id
FROM employees
GROUP BY department_id
HAVING MIN(salary) < (
   SELECT MIN(salary)
```

```
    FROM employees
    WHERE department_id = 50
);
```

No data was returned from this query.

8. Which statements are true about the subquery below?

```
SELECT employee_id, last_name
FROM employees
WHERE salary =
(SELECT MIN(salary)
FROM employees
GROUP BY department_id);
```

a. The inner query could be eliminated simply by changing the WHERE clause to WHERE MIN(salary).

b. The query wants the names of employees who make the same salary as the smallest salary in any department.

c. The query first selects the employee ID and last name, and then compares that to the salaries in every department.

d. This query will not execute.

b. The query wants the names of employees who make the same salary as the smallest salary in any department.

d. This query will not execute.

9. Write a pair-wise subquery listing the last_name, first_name, department_id, and manager_id for all employees that have the same department_ id and manager_id as employee 141. Exclude employee 141 from the result set.

```
SELECT last_name, first_name, department_id, manager_id
FROM employees
WHERE (department_id, manager_id) = (
    SELECT department_id, manager_id
    FROM employees
    WHERE employee_id = 141
)
AND employee_id <> 141;
```

10. Write a non-pair-wise subquery listing the last_name, first_name, department_id, and manager_id for all employees that have the same department_ id and manager_id as employee 141.

```
SELECT last_name, first_name, department_id, manager_id
FROM employees
WHERE department_id = (
    SELECT department_id
    FROM employees
    WHERE employee_id = 141
)
AND manager_id = (
```

```
    SELECT manager_id
    FROM employees
    WHERE employee_id = 141
);
```

Correlated Subqueries

1. Explain the main difference between correlated and non-correlated subqueries?

A non-correlated subquery is an independent query whose result is passed to the main query, whereas a correlated subquery is dependent on the outer query and is re-executed for each row processed by the outer query.

2. Write a query that lists the highest earners for each department. Include the last_name, department_id, and the salary for each employee.

```
SELECT last_name, department_id, salary
FROM employees outer
WHERE salary = (
   SELECT MAX(salary)
   FROM employees inner
   WHERE inner.department_id = outer.department_id
);
```

3. Examine the following select statement and finish it so that it will return the last_name, department_id, and salary of employees who have at least one person reporting to them. So we are effectively looking for managers only. In the partially written SELECT statement, the WHERE clause will work as it is. It is simply testing for the existence of a row in the subquery.

```
SELECT (enter columns here)
FROM (enter table name here) outer
WHERE 'x' IN (SELECT 'x'
FROM (enter table name here) inner
WHERE inner(enter column name here) = inner(enter column name here)
```

Finish off the statement by sorting the rows on the department_id column.

```
SELECT last_name, department_id, salary FROM employees outer WHERE EXISTS (SELECT 1
FROM employees inner WHERE inner.manager_id = outer.employee_id)ORDER BY department_id;
```

4. Using a WITH clause, write a SELECT statement to list the job_title of those jobs whose maximum salary is more than half the maximum salary of the entire company. Name your subquery MAX_CALC_SAL. Name the columns in the result JOB_TITLE and JOB_TOTAL, and sort the result on JOB_TOTAL in descending order.
Hint: Examine the jobs table. You will need to join JOBS and EMPLOYEES to display the job_title.

```
WITH MAX_CALC_SAL AS (SELECT job_title, MAX(salary) AS job_max_salary FROM
employees e JOIN jobs j ON e.job_id = j.job_id GROUP BY job_title)
SELECT job_title, job_max_salary AS JOB_TOTAL FROM MAX_CALC_SAL
WHERE job_max_salary > (SELECT MAX(salary) / 2 FROM employees)
ORDER BY JOB_TOTAL DESC;
```

Summarizing Queries for practice

INSERT Statements

Students should execute DESC tablename 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.

Correcting Errors: Data might be entered incorrectly, such as a wrong product price or an incorrect employee's designation. Altering the data helps maintain data accuracy and integrity.

2. 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.

```
-- DESC copy_d_cds;  -- To view the structure of the table

INSERT INTO copy_d_cds (cd_id, title, artist, genre, year) VALUES (1, 'The Best of Jazz',
'Various Artists', 'Jazz', 2023);
INSERT INTO copy_d_cds (cd_id, title, artist, genre, year) VALUES (2, 'Rock Legends',
'Various Artists', 'Rock', 2023);
INSERT INTO copy_d_cds (cd_id, title, artist, genre, year) VALUES (3, 'Pop Hits 2023',
'Various Artists', 'Pop', 2023);
INSERT INTO copy_d_cds (cd_id, title, artist, genre, year) VALUES (4,
'Classical Essentials', 'Various Artists', 'Classical', 2023);

-- Verify the insertion
SELECT * FROM copy_d_cds;
```

3. 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.

```
-- DESC copy_d_songs;  -- To view the structure of the table

INSERT INTO copy_d_songs
SELECT 1001, 'Football Anthem', 5, '03:45', 101 FROM dual
UNION ALL
SELECT 1002, 'Sixties Vibe', 6, '04:00', 102 FROM dual;

-- Verify the insertion
SELECT * FROM copy_d_songs;
```

4. Add the two new clients to the copy_d_clients table. Use either an implicit or an explicit INSERT.

```
-- DESC copy_d_clients;  -- To view the structure of the table

-- Explicit INSERT
INSERT INTO copy_d_clients (client_id, client_name, contact_number, email) VALUES
(1, 'John Doe', '123-456-7890', 'john@example.com');
INSERT INTO copy_d_clients (client_id, client_name, contact_number, email) VALUES
(2, 'Jane Smith', '987-654-3210', 'jane@example.com');

-- Verify the insertion
SELECT * FROM copy_d_clients;
```

5. Add the new client's events to the copy_d_events table. The cost of each event has not been determined at this date.

```
-- DESC copy_d_events;  -- To view the structure of the table

-- Explicit INSERT with NULL for cost
INSERT INTO copy_d_events (event_id, client_id, event_name, event_date, cost) VALUES
 (201, 1, 'Fall Football Party', '2024-09-23', NULL);
INSERT INTO copy_d_events (event_id, client_id, event_name, event_date, cost) VALUES
(202, 2, 'Sixties Theme Party', '2024-10-15', NULL);

-- Verify the insertion
SELECT * FROM copy_d_events;
```

6. 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)
);
```

Updating Column Values and Deleting Rows

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.

```
-- Assuming `item_name` and `price` are the columns in `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';
```

2. 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?)

```
-- Assuming `staff_name` and `overtime_pay` are the columns in `copy_f_staffs` table

-- Use NVL function to convert null to 0 for Bob Miller
UPDATE copy_f_staffs
SET overtime_pay = NVL(overtime_pay, 0) + 0.75
WHERE staff_name = 'Bob Miller';

UPDATE copy_f_staffs
SET overtime_pay = NVL(overtime_pay, 0) + 0.85
WHERE staff_name = 'Sue Doe';
```

3. Add the orders shown to the Global Fast Foods copy_f_orders table:
ORDER_NUMBER ORDER_DATE ORDER_TOTAL CUST_ID STAFF_ID
5680 June 12, 2004 159.78 145 9
5691 09-23-2004 145.98 225 12
5701 July 4, 2004 229.31 230 12

```
-- Assuming the columns are `order_number`, `order_date`, `order_total`, `cust_id`, and `staff_id`

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);
```

4. 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?
ID FIRST_
NAME
LAST_
NAME
ADDRESS CITY STATE ZIP PHONE_NUMBER
145 Katie Hernandez 92 Chico
Way Los Angeles CA 98008 8586667641
225 Daniel Spode 1923 Silverado
Denver CO 80219 7193343523
230 Adam Zurn 5 Admiral Way Seattle WA 4258879009

```
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');
```

5. 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.

```
-- Assuming `salary` is the column in `copy_f_staffs` table

UPDATE copy_f_staffs
SET salary = (SELECT salary FROM copy_f_staffs WHERE staff_name = 'Bob Miller')
WHERE staff_name = 'Sue Doe';
```

6. 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 at right.
ID FIRST_NAME LAST_NAME BIRTHDATE SALARY STAFF
TYPE
25 Kai Kim 3-Nov-1988 6.75 Order Taker

```
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');
```

7. 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 staff_name = 'Sue Doe'),
   overtime_pay = 0
WHERE id = 25;
```

8. Execute the following SQL statement. Record your results.
DELETE from departments
WHERE department_id = 60;

DELETE FROM departments
WHERE department_id = 60;

-- Verify the deletion
SELECT * FROM departments WHERE department_id = 60;

9. 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 id = 25;

-- Verify the deletion
SELECT * FROM copy_f_staffs WHERE id = 25;

10. 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 employee_id IN (SELECT employee_id FROM job_history);
SELECT * FROM lesson7_emp;

DEFAULT Values, MERGE, and Multi-Table Inserts

1. When would you want a DEFAULT value?

A DEFAULT value is useful when you want to ensure that a column has a predefined value if no Specific value is provided during an insert.

2. 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:
a. 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;
```

b. Alter the current START_DATE column attributes using:

```
ALTER TABLE copy_f_promotional_menus
MODIFY (START_DATE DEFAULT SYSDATE);
```

c. 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');
SELECT * FROM copy_f_promotional_menus WHERE promotion_code = 120;
```

3. 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 inventory in an hour, but she doesn't want the original D_CDS table changed. Prepare an updated inventory list just for her.
a. Assign new cd_numbers to each new CD acquired.

```
CREATE TABLE manager_copy_d_cds AS
SELECT * FROM D_CDS;
```

b. 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;
```

c. 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, publisher, year)
VALUES (20, 'Hello World Here I Am', 'Middle Earth Records', '1998');
SELECT * FROM manager_copy_d_cds WHERE cd_number = 20;
```

d. 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 mc
USING D_CDS d
ON (mc.cd_number = d.cd_number)
WHEN MATCHED THEN
  UPDATE SET mc.title = d.title, mc.year = d.year
WHEN NOT MATCHED THEN
  INSERT (cd_number, title, publisher, year)
  VALUES (d.cd_number, d.title, d.publisher, d.year);

-- Verify the merge
SELECT * FROM manager_copy_d_cds;
```

4. 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

Creating Tables

1. Complete the GRADUATE CANDIDATE table instance chart. Credits is a foreign-key column referencing the requirements table.
2. Write the syntax to create the grad_candidates table.

```
CREATE TABLE grad_candidates (
  candidate_id NUMBER PRIMARY KEY,
  first_name VARCHAR2(50),
  last_name VARCHAR2(50),
  credits NUMBER,
  FOREIGN KEY (credits) REFERENCES requirements(credits));
```

3. Confirm creation of the table using DESCRIBE.

```
DESC grad_candidates;
```

4. 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;
```

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

```
INSERT INTO smith_table (candidate_id, first_name, last_name, credits)
VALUES (1, 'Smith', 'Joe', 30);
SELECT * FROM smith_table;
```

6. 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.

```
-- USER_TABLES: Returns a list of tables owned by the user
SELECT * FROM USER_TABLES;

-- USER_OBJECTS: Returns a list of objects (tables, indexes, views, etc.) owned by the user
SELECT * FROM USER_OBJECTS;

-- USER_CATALOG or USER_CAT: Returns a list of user-owned tables, views, synonyms,
 and sequences
SELECT * FROM USER_CATALOG;
```

Modifying a Table

Before beginning the practice exercises, execute a DESCRIBE for each of the following tables: o_employees and o_jobs. These tables will be used in the exercises. You will need to know which columns do not allow null values.
NOTE: If students have not already created the o_employees, o_departments, and o_jobs tables they should create them using the four steps outlined in the practice.
1. Create the three o_tables – jobs, employees, and departments – using the syntax:

```
CREATE TABLE o_jobs (
  job_id NUMBER PRIMARY KEY,
  job_title VARCHAR2(50) NOT NULL
);

CREATE TABLE o_departments (
  department_id NUMBER PRIMARY KEY,
  department_name VARCHAR2(50) NOT NULL
);

CREATE TABLE o_employees (
  employee_id NUMBER PRIMARY KEY,
  first_name VARCHAR2(50) NOT NULL,
  last_name VARCHAR2(50) NOT NULL,
  job_id NUMBER,
  department_id NUMBER,
  FOREIGN KEY (job_id) REFERENCES o_jobs(job_id),
  FOREIGN KEY (department_id) REFERENCES o_departments(department_id)
);
```

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

```
INSERT INTO o_jobs (job_id, job_title)
VALUES (10, 'Human Resources');
```

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

```
INSERT INTO o_employees (employee_id, first_name, last_name, job_id, department_id)
VALUES (1, 'Alice', 'Smith', 10, 1);

INSERT INTO o_employees (employee_id, first_name, last_name, job_id, department_id)
VALUES (2, 'Bob', 'Johnson', 10, 1);

INSERT INTO o_employees (employee_id, first_name, last_name, job_id, department_id)
VALUES (3, 'Charlie', 'Brown', 10, 1);
```

4. Add Human Resources to the departments table:

```
INSERT INTO o_departments (department_id, department_name)
VALUES (1, 'Human Resources');
```

5. Why is it important to be able to modify a table?

Update Schema: Reflect changes in business requirements.
Add Constraints: Enhance data integrity and enforce business rules.

1. CREATE a table called Artists.
a. Add the following to the table:
• artist ID
• first name
• last name
• band name
• email
• hourly rate
• song ID from d_songs table

```
CREATE TABLE Artists(
  artist_id NUMBER PRIMARY KEY,
  first_name VARCHAR2(50),
  last_name VARCHAR2(50),
  band_name VARCHAR2(100),
  email VARCHAR2(100),
  hourly_rate NUMBER,
  song_id NUMBER,
  FOREIGN KEY (song_id) REFERENCES d_songs(song_id)
);
```

b. INSERT one artist from the d_songs table

```
INSERT INTO Artists (artist_id, first_name, last_name, band_name, email, hourly_rate, song_id)
SELECT 1, 'John', 'Doe', 'The Band', 'john@example.com', 100, song_id
FROM d_songs
WHERE song_id = 1;
```

c. INSERT one artist of your own choosing; leave song_id blank.

```
INSERT INTO Artists (artist_id, first_name, last_name, band_name, email, hourly_rate, song_id)
VALUES (2, 'Jane', 'Smith', 'Solo Artist', 'jane@example.com', 120, NULL);
```

d. Give an example how each of the following may be used on the table that you have created:
1) ALTER TABLE
2) DROP TABLE
3) RENAME TABLE
4) TRUNCATE
5) COMMENT ON TABLE

```
ALTER TABLE Artists ADD (genre VARCHAR2(50));
DROP TABLE Artists;
RENAME Artists TO Musicians;
TRUNCATE TABLE Musicians;
COMMENT ON TABLE Musicians IS 'This table contains musician details';
```

2. In your o_employees table, enter a new column called “Termination.” The datatype for the new column 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(100) DEFAULT TO_CHAR(SYSDATE, 'Month DDth, YYYY'));
```

3. 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);
```

4. Truncate the o_jobs table. Then do a SELECT * statement. Are the columns still there? Is the data still there?

```
TRUNCATE TABLE o_jobs;

-- Verify the truncation
SELECT * FROM o_jobs;
```

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

TRUNCATE: Removes all rows from a table without logging individual row deletions. Cannot be rolled back. The structure of the table remains.
DELETE: Removes rows one by one and logs each deletion. Can be rolled back. The structure of the table remains.
DROP: Deletes the table structure and its data permanently. Cannot be rolled back.

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

Changes that can be made:

Change Data Type: If no data loss will occur.
Add Constraints: Such as NOT NULL or UNIQUE.
Set/Change Default Value: To ensure a predefined value for new rows.
Changes that cannot be made:

Decrease Size: If data exists that exceeds the new size.
Change to Incompatible Data Type: Such as from VARCHAR2 to NUMBER if incompatible data exists.

7. 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';

-- View the comment
```

```
SELECT * FROM USER_TAB_COMMENTS WHERE TABLE_NAME = 'O_JOBS';
```

8. Rename the o_jobs table to o_job_description.

```
RENAME o_jobs TO o_job_description;

-- Verify the renaming
DESC o_job_description;
```

9.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 a copy of `f_staffs` table
CREATE TABLE copy_f_staffs AS
SELECT * FROM f_staffs;
```

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

```
-- Describe the new table
DESC copy_f_staffs;
```

C.Drop the table.

```
-- Drop the table
DROP TABLE copy_f_staffs;
```

D.Try to select from the table.

```
-- Try to select from the dropped table (will cause an error)
SELECT * FROM copy_f_staffs;
```

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

```
SELECT * FROM USER_RECYCLEBIN;
```

a. 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==$0, you need to write query that refers to "BIN$Q+x1nJdcUnngQESYELVIdQ==$0".

```
-- Select from the dropped table using the recycle bin name
SELECT * FROM "BIN$Q+x1nJdcUnngQESYELVIdQ==$0";
```

b. Undrop the table.

```
-- Undrop the table
FLASHBACK TABLE copy_f_staffs TO BEFORE DROP;
```

c. Describe the table.

```
-- Describe the table
```

DESC 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;

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

c. Issue a select statement to see all rows and all columns from the copy_f_staffs table;

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

e. Issue a select statement to see all rows and all columns from the copy_f_staffs table;

f. Now, issue a FLASHBACK QUERY statement against the copy_f_staffs table, so you can see all the changes made.

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.

```
-- Issue a select statement
SELECT * FROM copy_f_staffs;

-- Update Sue Doe's salary
UPDATE copy_f_staffs SET salary = 12 WHERE staff_name = 'Sue Doe';
COMMIT;

-- Verify the update
SELECT * FROM copy_f_staffs;

-- Update Sue Doe's salary again
UPDATE copy_f_staffs SET salary = 2 WHERE staff_name = 'Sue Doe';
COMMIT;

-- Verify the update
SELECT * FROM copy_f_staffs;

-- Flashback query to see all changes made
SELECT * FROM copy_f_staffs AS OF TIMESTAMP (SYSTIMESTAMP - INTERVAL
 '10' MINUTE);

-- Update Sue Doe's salary back to the original value
UPDATE copy_f_staffs SET salary = original_salary WHERE staff_name = 'Sue Doe';
```