RAJALAKSHMI ENGINEERING COLLEGE RAJALAKSHMI NAGAR, THANDALAM – 602 105

RAJALAKSHMI
ENGINEERING COLLEGE
An AUTONOMOUS Institution
Affiliated to ANNA UNIVERSITY, Chennai

CS19443

DATABASE MANAGEMENT SYSTEMS LAB

Laboratory
Record Note Book

Name

:……. ……... Year / Branch / Section :

……….. ………………………………. University Register No. : ……………..

…………………………………… College Roll No. :

……………………..……………………………… Semester :

………………………………..…………………………………… Academic

Year : ……………………………………………………….

INDEX

Name : ____________________ **Branch :** __________ **Sec :** ________ **Roll No :** ____________

S.No.	Date	Title	Page No.	Teacher's Sign / Remarks

EXERCISE-1

Creating and Managing Tables

1. Create the DEPT table based on the DEPARTMENT following the table instance chart below. Confirm that the table is created.

Column name	ID	NAME
Key Type		
Nulls/Unique		
FK table		
FK column		
Data Type	Number	Varchar2
Length	7	25

```
CREATE TABLE DEPT (
ID NUMBER(7),
NAME VARCHAR2(25),
CONSTRAINT PK_DEPT PRIMARY KEY (ID)
);
DESCRIBE DEPT;
```

2. Create the EMP table based on the following instance chart. Confirm that the table is created.

Column name	ID	LAST_NAME	FIRST_NAME	DEPT_ID
Key Type				
Nulls/Unique				
FK table				
FK column				
Data Type	Number	Varchar2	Varchar2	Number
Length	7	25	25	7

```
CREATE TABLE EMP (
ID NUMBER(7),
LAST_NAME VARCHAR2(25),
FIRST_NAME VARCHAR2(25),
 DEPT_ID NUMBER(7),
CONSTRAINT PK_EMP PRIMARY KEY (ID),
CONSTRAINT FK_EMP_DEPT FOREIGN KEY (DEPT_ID) REFERENCES DEPT(ID)
);
Confirm the table creation
DESCRIBE EMP;
```

3. Modify the EMP table to allow for longer employee last names. Confirm the modification.(Hint: Increase the size to 50)

ALTER TABLE EMP MODIFY (LAST_NAME VARCHAR2(50));

Confirm the modification

DESCRIBE EMP

4. Create the EMPLOYEES2 table based on the structure of EMPLOYEES table. Include Only the Employee_id, First_name, Last_name, Salary and Dept_id coloumns. Name the columns Id, First_name, Last_name, salary and Dept_id respectively.

```
CREATE TABLE EMPLOYEES2 AS SELECT
   ID AS Id,
   FIRST_NAME AS First_name,
   LAST_NAME AS Last_name,
   SALARY,
   DEPT_ID AS Dept_id
FROM EMPLOYEES;
```

Confirm the table creation

DESCRIBE EMPLOYEES2;

5. Drop the EMP table.

DROP TABLE EMP;

6. Rename the EMPLOYEES2 table as EMP.

```
RENAME TABLE EMPLOYEES2 TO EMP;
```

7. Add a comment on DEPT and EMP tables. Confirm the modification by describing the table.

```
ALTER TABLE DEPT COMMENT = 'Department Table';

ALTER TABLE EMP COMMENT = 'Employee Table';

DESCRIBE DEPT;

DESCRIBE EMP;
```

8. Drop the First_name column from the EMP table and confirm it.

```
ALTER TABLE EMP DROP COLUMN FIRST_NAME;
```

Confirm the column has been dropped

```
DESCRIBE EMP;
```

EXERCISE-2

MANIPULATING DATA

Find the Solution for the following:

1. Create MY_EMPLOYEE table with the following structure

NAME	NULL?	TYPE
ID	Not null	Number(4)
Last_name		Varchar(25)
First_name		Varchar(25)
Userid		Varchar(25)
Salary		Number(9,2)

2. Add the first and second rows data to MY_EMPLOYEE table from the following sample data.

ID	**Last_name**	**First_name**	**Userid**	**salary**
1	Patel	Ralph	rpatel	895
2	Dancs	Betty	bdancs	860
3	Biri	Ben	bbiri	1100
4	Newman	Chad	Cnewman	750
5	Ropebur	Audrey	aropebur	1550

3. Display the table with values.

SELECT * FROM MY_EMPLOYEE;

4. Populate the next two rows of data from the sample data. Concatenate the first letter of the first_name with the first seven characters of the last_name to produce Userid.

INSERT INTO MY_EMPLOYEE (ID, Last_name, First_name, Userid, Salary)

VALUES

(3, 'Biri', 'Ben', CONCAT(LEFT('Ben', 1), LEFT('Biri', 7)), 1100),

(4, 'Newman', 'Chad', CONCAT(LEFT('Chad', 1), LEFT('Newman', 7)), 750);

Confirm the data insertion

SELECT * FROM MY_EMPLOYEE;

5. Make the data additions permanent.

COMMIT;

6. Change the last name of employee 3 to Drexler.

INSERT INTO MY_EMPLOYEE (ID, Last_name, First_name, Userid, Salary)

VALUES

(3, 'Biri', 'Ben', CONCAT(LEFT('Ben', 1), LEFT('Biri', 7)), 1100),

(4, 'Newman', 'Chad', CONCAT(LEFT('Chad', 1), LEFT('Newman', 7)), 750);

Confirm the data insertion

SELECT * FROM MY_EMPLOYEE;

7. Change the salary to 1000 for all the employees with a salary less than 900.

UPDATE MY_EMPLOYEE

SET Salary = 1000

WHERE Salary < 900;

8. Delete Betty dancs from MY _EMPLOYEE table.

DELETE FROM MY_EMPLOYEE

WHERE Last_name = 'Dancs' AND First_name = 'Betty';

PRACTICE QUESTIONS

Working with Columns, Characters, and Rows

1. The manager of Global Fast Foods would like to send out coupons for the upcoming sale. He wants to send one coupon to each household. Create the SELECT statement that returns the customer last name and a mailing address.

```
SELECT last_name, address FROM customers;
```

2. Each statement below has errors. Correct the errors and execute the query in Oracle Application Express.

a. SELECT first name FROM f_staffs;

```
SELECT first_name FROM f_staffs;
```

b. SELECT first_name |" " | last_name AS "DJs on Demand Clients" FROM d_clients;

```
SELECT CONCAT(first_name, ' ', last_name) AS "DJs on Demand Clients"
FROM d_clients;
```

c. SELECT DISCTINCT f_order_lines FROM quantity;

```
SELECT DISTINCT quantity FROM f_order_lines;
```

d. SELECT order number FROM f_orders;

```
SELECT order_number FROM f_orders;
```

3. Sue, Bob, and Monique were the employees of the month. Using the f_staffs table, create a SELECT statement to display the results as shown in the Super Star chart.

Super Star
*** Sue *** Sue ***
*** Bob *** Bob ***
*** Monique *** Monique ***

```
SELECT CONCAT('*** ', first_name, ' *** ', first_name, ' ***') AS "Super Star" FROM
f_staffs
WHERE first_name IN ('Sue', 'Bob', 'Monique');
```

4.Which of the following is TRUE about the following query?

SELECT first_name, DISTINCT birthdate
FROM f_staffs;
a. Only two rows will be returned.
b. Four rows will be returned.
c. Only Fred 05-Jan-1988 and Lizzie 10-Nov-1987 will be returned.
d. No rows will be returned.

b. Four rows will be returned.

5. Global Fast Foods has decided to give all staff members a 5% raise. Prepare a report that presents the output as shown in the chart.

EMPLOYEE LAST NAME	CURRENT SALARY	SALARY WITH 5% RAISE

```
SELECT last_name AS "EMPLOYEE LAST NAME", salary AS "CURRENT SALARY",
salary * 1.05 AS
"SALARY WITH 5% RAISE"FROM f_staffs;
```

6. Create a query that will return the structure of the Oracle database EMPLOYEES table. Which columns are marked “nullable”? What does this mean?

```
SHOW COLUMNS FROM EMPLOYEES;
```

7. The owners of DJs on Demand would like a report of all items in their D_CDs table with the following column headings: Inventory Item, CD Title, Music Producer, and Year Purchased. Prepare this report.
8.True/False – The following SELECT statement executes successfully:
SELECT last_name, job_id, salary AS Sal FROM employees;

```
SELECT inventory_item AS "Inventory Item",
    cd_title AS "CD Title",
    music_producer AS "Music Producer",
    year_purchased AS "Year Purchased"
FROM d_cds;
```

8.True/False – The following SELECT statement executes successfully:
SELECT
last_name, job_id, salary AS Sal FROM employees;

True

9.True/False – The following SELECT statement executes successfully:
SELECT * FROM job_grades;

True

10.There are four coding errors in this statement. Can you identify them?

SELECT employee_id, last_name sal x 12 ANNUAL SALARY FROM employees;
SELECT employee_id,last_name, salary * 12 AS "ANNUAL SALARY"
FROM employees;

11.In the arithmetic expression salary*12 - 400, which operation will be evaluated first?

Multiplication will be evaluated first.

12. Which of the following can be used in the SELECT statement to return all columns of data in the Global Fast Foods f_staffs table?
a. column names
b. *
c. DISTINCT id
d. both a and b

b. *

13. Using SQL to choose the columns in a table uses which capability?
a. selection
b. projection
c. partitioning
d. join

b. projection

14. SELECT last_name AS "Employee". The column heading in the query result will appear as: a. EMPLOYEE
b. employee
c. Employee
d. "Employee:

c. Employee

15. Which expression below will produce the largest value?
a. SELECT salary*6 + 100
b. SELECT salary* (6 + 100)
c. SELECT 6(salary+ 100)
d. SELECT salary+6*100

b. SELECT salary * (6 + 100)

16. Which statement below will return a list of employees in the following format? Mr./Ms. Steven King is an employee of our company.
a. SELECT "Mr./Ms."||first_name||' '||last_name 'is an employee of our company.' AS "Employees" FROM employees;
b. SELECT 'Mr./Ms. 'first_name,last_name ||' '||'is an employee of our company.' FROM employees;
c. SELECT 'Mr./Ms. '||first_name||' '||last_name ||' '||'is an employee of our company.' AS "Employees" FROM employees ;
d. SELECT Mr./Ms. ||first_name||' '||last_name ||' '||"is an employee of our company." AS "Employees" FROM employees

SELECT CONCAT('Mr./Ms. ', first_name, ' ', last_name, ' is an employee of our company.') AS "Employees"
FROM employees;

17. Which is true about SQL statements?
a. SQL statements are case-sensitive
b. SQL clauses should not be written on separate lines.
c. Keywords cannot be abbreviated or split across lines.
d. SQL keywords are typically entered in lowercase; all other words in uppercase.

c. Keywords cannot be abbreviated or split across lines.

18. Which queries will return three columns each with UPPERCASE column headings? a. SELECT "Department_id", "Last_name", "First_name"
FROM employees;
b. SELECT DEPARTMENT_ID, LAST_NAME, FIRST_NAME
FROM employees;
c. SELECT department_id, last_name, first_name AS UPPER CASE
FROM employees
d. SELECT department_id, last_name, first_name
FROM employees;

b. SELECT DEPARTMENT_ID, LAST_NAME, FIRST_NAME FROM employees;

19. Which statement below will likely fail?

a. SELCT * FROM employees;
b. Select * FROM employees;
c. SELECT * FROM EMPLOYEES;
d. SelecT* FROM employees;

a. SELCT * FROM employees;

EXERCISE-3

INCLUDING CONSTRAINTS

Find the Solution for the following:

1. Add a table-level PRIMARY KEY constraint to the EMP table on the ID column.The constraint should be named at creation. Name the constraint my_emp_id_pk.

```
ALTER TABLE EMP

ADD CONSTRAINT my_emp_id_pk PRIMARY KEY (ID);
```

2. Create a PRIMARY KEY constraint to the DEPT table using the ID column. The constraint should be named at creation. Name the constraint my_dept_id_pk.

```
ALTER TABLE DEPT

ADD CONSTRAINT my_dept_id_pk PRIMARY KEY (ID);
```

3. Add a column DEPT_ID to the EMP table. Add a foreign key reference on the EMP table that ensures that the employee is not assigned to non-existent department. Name the constraint my_emp_dept_id_fk.

```
ALTER TABLE EMP
ADD DEPT_ID INT;

ALTER TABLE EMP
ADD CONSTRAINT my_emp_dept_id_fk FOREIGN KEY (DEPT_ID) REFERENCES
DEPT(ID);
```

4. Modify the EMP table. Add a COMMISSION column of NUMBER data type, precision 2, scale 2. Add a constraint to the commission column that ensures that a commission value is greater than zero.

```
ALTER TABLE EMP
ADD COMMISSION DECIMAL(2, 2);

ALTER TABLE EMP
ADD CONSTRAINT chk_commission_positive CHECK (COMMISSION > 0);
```

PRACTICE QUESTIONS

Limit Rows Selected

1. Using the Global Fast Foods database, retrieve the customer's first name, last name, and address for the customer who uses ID 456.

```
SELECT first_name, last_name, address

FROM customers

WHERE id = 456;
```

2. Show the name, start date, and end date for Global Fast Foods' promotional item "ballpen and highlighter" giveaway.

```
SELECT name, start_date, end_date

FROM promotions

WHERE name = 'ballpen and highlighter';
```

3. Create a SQL statement that produces the following output:

Oldest
The 1997 recording in our database is The Celebrants Live in Concert

```
SELECT 'Oldest' AS "Oldest",
     'The 1997 recording in our database is The Celebrants Live in Concert' AS "Description"
FROM dual;
```

4. The following query was supposed to return the CD title "Carpe Diem" but no rows were returned. Correct the mistake in the statement and show the output.

```
SELECT produce, title
FROM d_cds
WHERE title = 'carpe diem' ;
SELECT produce, title
FROM d_cds
WHERE title = 'Carpe Diem';
```

5. The manager of DJs on Demand would like a report of all the CD titles and years of CDs that were produced before 2000.

```
SELECT title, year
FROM d_cds
WHERE year < 2000;
```

6. Which values will be selected in the following query?

```
SELECT salary
FROM employees
WHERE salary < = 5000;
```

a. 5000
b. 0 - 4999
c. 2500
d. 5

b. 0 - 4999

7. Write a SQL statement that will display the student number (studentno), first name (fname), and last name (lname) for all students who are female (F) in the table named students.

```
SELECT studentno, fname, lname
FROM students
WHERE gender = 'F';
```

8. Write a SQL statement that will display the student number (studentno) of any student who has a PE major in the table named students. Title the studentno column Student Number.

```
SELECT studentno AS "Student Number"
FROM students
WHERE major = 'PE';
```

9. Write a SQL statement that lists all information about all male students in the table named students.

```
SELECT *
FROM students
WHERE gender = 'M';
```

10. Write a SQL statement that will list the titles and years of all the DJs on Demand's CDs that were not produced in 2000.

```
SELECT title, year
FROM d_cds
WHERE year <> 2000;
```

11. Write a SQL statement that lists the Global Fast Foods employees who were born before 1980.

```
SELECT *
FROM f_staffs
WHERE birthdate < '1980-01-01';
```

EXERCISE-4

Writing Basic SQL SELECT Statements

Find the Solution for the following:

True OR False

1. The following statement executes successfully.

Identify the Errors

```
SELECT employee_id, last_name
sal*12 ANNUAL SALARY
FROM employees;
```

Queries

False

```
SELECT employee_id, last_name,
      sal * 12 AS "ANNUAL SALARY"
FROM employees;
```

2. Show the structure of departments the table. Select all the data from it.

```
-- Show the structure of the departments table
DESCRIBE departments;

-- Select all the data from the departments table
SELECT * FROM departments;
```

3. Create a query to display the last name, job code, hire date, and employee number for each employee, with employee number appearing first.

```
SELECT employee_number, last_name, job_code, hire_date
FROM employees;
```

4. Provide an alias STARTDATE for the hire date.

```
SELECT employee_number, last_name, job_code, hire_date AS STARTDATE
FROM employees;
```

5. Create a query to display unique job codes from the employee table.

```
SELECT DISTINCT job_code
FROM employees;
```

6. Display the last name concatenated with the job ID , separated by a comma and space, and name the column EMPLOYEE and TITLE.

```
SELECT CONCAT(last_name, ', ', job_id) AS "EMPLOYEE and TITLE"
FROM employees;
```

7. Create a query to display all the data from the employees table. Separate each column by a comma. Name the column THE_OUTPUT.

```
SELECT
    CONCAT_WS(', ', employee_id, first_name, last_name, email, phone_number, hire_date, job_id, salary, commission_pct, manager_id, department_id) AS THE_OUTPUT
FROM employees;
```

Practice Questions

COMPARISON OPERATORS

1. Who are the partners of DJs on Demand who do not get an authorized expense amount?

SELECT partner_name FROM partners WHERE authorized_expense_amount IS NULL;

2. Select all the Oracle database employees whose last names end with “s”. Change the heading of the column to read Possible Candidates.

SELECT last_name AS "Possible Candidates" FROM employees WHERE last_name LIKE '%s';

3. Which statement(s) are valid?

a. WHERE quantity <> NULL;

b. WHERE quantity = NULL;

c. WHERE quantity IS NULL;

d. WHERE quantity != NULL;

c. WHERE quantity IS NULL;

4. Write a SQL statement that lists the songs in the DJs on Demand inventory that are type code 77, 12, or 1.

SELECT song_title

FROM d_songs

WHERE type_code IN (77, 12, 1);

Logical Comparisons and Precedence Rules

1. Execute the two queries below. Why do these nearly identical statements produce two different results? Name the difference and explain why.

```
SELECT code, description
FROM d_themes
WHERE code >200 AND description IN('Tropical', 'Football', 'Carnival');
SELECT  code, description
FROM d_themes
 WHERE code >200 OR description IN('Tropical', 'Football', 'Carnival');
```

```
-- Query 1
SELECT code, description FROM d_themes WHERE code > 200 AND description IN
('Tropical','Football', 'Carnival');
Query 2
SELECT code, description FROM d_themes WHERE code > 200 OR description IN ('Tropical',
'Football',
'Carnival');
```

2. Display the last names of all Global Fast Foods employees who have “e” and “i” in their last names.

```
SELECT last_name
FROM employees
WHERE last_name LIKE '%e%' AND last_name LIKE '%i%';
```

3. “I need to know who the Global Fast Foods employees are that make more than $6.50/hour and their position is not order taker.”

```
SELECT first_name, last_name
FROM employees
WHERE salary > 6.50 AND position != 'order taker';
```

4. Using the employees table, write a query to display all employees whose last names start with "D" and have "a" and "e" anywhere in their last name.

```
SELECT *FROM employees
WHERE last_name LIKE 'D%' AND last_name LIKE '%a%' AND last_name LIKE '%e%';
```

5. In which venues did DJs on Demand have events that were not in private homes?

```
SELECT * FROM employees
WHERE last_name LIKE 'D%' AND last_name LIKE '%a%' AND last_name LIKE '%e%';
```

6. Which list of operators is in the correct order from highest precedence to lowest precedence?

a.AND, NOT, OR

b. NOT, OR, AND

c. NOT, AND, OR

c. NOT, AND, OR

For questions 7 and 8, write SQL statements that will produce the desired output.

7. Who am I?

I was hired by Oracle after May 1998 but before June of 1999. My salary is less than $8000 per month, and I have an “en” in my last name.

```
SELECT first_name, last_nam FROM employees WHERE hire_date > '1998-05-31' AND
hire_date <     '1999-06-01' AND salary < 8000 AND last_name LIKE '%en%';
```

8. What's my email address?

Because I have been working for Oracle since the beginning of 1996, I make more than $9000 per month. Because I make so much money, I don't get a commission

```
SELECT email
FROM employees
WHERE hire_date >= '1996-01-01' AND salary > 9000 AND commission_pct IS NULL;
```

EXERCISE-5

Restricting and Sorting data

Find the Solution for the following:

1. Create a query to display the last name and salary of employees earning more than 12000.

```
SELECT last_name, salary FROM employees WHERE salary > 12000;
```

2. Create a query to display the employee last name and department number for employee number 176.

```
SELECT last_name, department_id

FROM employees

WHERE employee_id = 176;
```

3. Create a query to display the last name and salary of employees whose salary is not in the range of 5000 and 12000. (hints: not between)

```
SELECT last_name, salary

FROM employees

WHERE salary NOT BETWEEN 5000 AND 12000;
```

4. Display the employee last name, job ID, and start date of employees hired between February 20,1998 and May 1,1998.order the query in ascending order by start date.(hints: between)

```
SELECT last_name, job_id, hire_date
FROM employees
WHERE hire_date BETWEEN '1998-02-20' AND '1998-05-01'
ORDER BY hire_date ASC;
```

5. Display the last name and department number of all employees in departments 20 and 50 in alphabetical order by name.(hints: in, orderby)

```
SELECT last_name, department_id
FROM employees
WHERE department_id IN (20, 50)
ORDER BY last_name;
```

6. Display the last name and salary of all employees who earn between 5000 and 12000 and are in departments 20 and 50 in alphabetical order by name. Label the columns EMPLOYEE, MONTHLY SALARY respectively.(hints: between, in)

```
SELECT last_name AS EMPLOYEE, salary AS "MONTHLY SALARY"
FROM employees
WHERE salary BETWEEN 5000 AND 12000 AND department_id IN (20, 50)
ORDER BY last_name;
```

7. Display the last name and hire date of every employee who was hired in 1994.(hints: like)

```
SELECT last_name, hire_date
FROM employees
WHERE hire_date LIKE '1994%';
```

8. Display the last name and job title of all employees who do not have a manager.(hints: is null)

```
SELECT last_name, job_id
FROM employees
WHERE manager_id IS NULL;
```

9. Display the last name, salary, and commission for all employees who earn commissions. Sort data in descending order of salary and commissions.(hints: is not null,orderby)

```
SELECT last_name, salary, commission_pct
FROM employees
WHERE commission_pct IS NOT NULL
ORDER BY salary DESC, commission_pct DESC;
```

10. Display the last name of all employees where the third letter of the name is ***a***.(hints:like)

```
SELECT last_name
FROM employees
WHERE last_name LIKE '__a%';
```

11. Display the last name of all employees who have an a and an ***e*** in their last name.(hints: like)

```
SELECT last_name FROM employees
WHERE last_name LIKE '%a%' AND last_name LIKE '%e%';
```

12. Display the last name and job and salary for all employees whose job is sales representative or stock clerk and whose salary is not equal to 2500 ,3500 or 7000.(hints:in,not in)

```
SELECT last_name, job_id, salary FROM employees
WHERE job_id IN ('SA_REP', 'ST_CLERK') AND salary NOT IN (2500, 3500, 7000);
```

13. Display the last name, salary, and commission for all employees whose commission amount is 20%.(hints:use predicate logic)

```
SELECT last_name, salary, commission_pct
FROM employees
WHERE commission_pct = 0.20;
```

Practice Questions

Sorting rows

1. In the example below, assign the employee_id column the alias of “Number.” Complete the SQL statement to order the result set by the column alias.

SELECT employee_id, first_name, last_name FROM employees;

SELECT employee_id AS "Number", first_name, last_name FROM employees ORDER BY "Number";

2. Create a query that will return all the DJs on Demand CD titles ordered by year with titles in alphabetical order by year.

SELECT title FROM d_cds ORDER BY year, title;

3. Order the DJs on Demand songs by descending title. Use the alias "Our Collection" for the song title.

SELECT song_title AS "Our Collection" FROM d_songs ORDER BY "Our Collection" DESC;

4. Write a SQL statement using the ORDER BY clause that could retrieve the information needed.

SELECT employee_id, first_name, last_name, hire_date FROM employees ORDER BY last_name ASC, hire_date DESC;

EXERCISE-6

Single Row Functions

Find the Solution for the following:

1. Write a query to display the current date. Label the column Date.

```
SELECT SYSDATE AS "Date" FROM DUAL;
```

2. The HR department needs a report to display the employee number, last name, salary, and increased by 15.5% (expressed as a whole number) for each employee. Label the column New Salary.

```
SELECT employee_id, last_name, salary, ROUND(salary * 1.155) AS "New Salary" FROM employees;
```

3. Modify your query lab_03_02.sql to add a column that subtracts the old salary from the new salary. Label the column Increase.

```
SELECT employee_id, last_name, salary, ROUND(salary * 1.155) AS "New Salary",
ROUND(salary * 1.155) - salary AS "Increase" FROM employees;
```

4. Write a query that displays the last name (with the first letter uppercase and all other letters lowercase) and the length of the last name for all employees whose name starts with the letters J, A, or M. Give each column an appropriate label. Sort the results by the employees‘ last names.

```
SELECT INITCAP(last_name) AS "Last Name", LENGTH(last_name) AS "Length" FROM
employees WHERE last_name LIKE 'J%' OR last_name LIKE 'A%' OR last_name LIKE 'M%'
ORDER BY last_name;
```

5. Rewrite the query so that the user is prompted to enter a letter that starts the last name. For example, if the user enters H when prompted for a letter, then the output should show all employees whose last name starts with the letter H.

```
SELECT INITCAP(last_name) AS "Last Name", LENGTH(last_name) AS "Length" FROM
employees WHERE last_name LIKE '&enter_letter%' ORDER BY last_name;
```

6. The HR department wants to find the length of employment for each employee. For each employee, display the last name and calculate the number of months between today and the date on which the employee was hired. Label the column MONTHS_WORKED. Order your results by the number of months employed. Round the number of months up to the closest whole number.

```
SELECT last_name, CEIL(MONTHS_BETWEEN(SYSDATE, hire_date)) AS
"MONTHS_WORKED" FROM employees ORDER BY "MONTHS_WORKED";
```

Note: Your results will differ.

7. Create a report that produces the following for each employee:
<employee last name> earns <salary> monthly but wants <3 times salary>. Label the column Dream Salaries.

```
SELECT last_name || ' earns ' || salary || ' monthly but wants ' || salary * 3 AS "Dream Salaries"
FROM employees;
```

8. Create a query to display the last name and salary for all employees. Format the salary to be 15 characters long, left-padded with the $ symbol. Label the column SALARY.

SELECT last_name, LPAD(salary, 15, '$') AS "SALARY" FROM employees;

9. Display each employee‘s last name, hire date, and salary review date, which is the first Monday after six months of service. Label the column REVIEW. Format the dates to appear in the format similar to ―Monday, the Thirty-First of July, 2000.‖

SELECT last_name, hire_date, TO_CHAR(NEXT_DAY(ADD_MONTHS(hire_date, 6), 'MONDAY'), 'Day, "the" DD "of" Month, YYYY') AS "REVIEW" FROM employees;

10. Display the last name, hire date, and day of the week on which the employee started. Label the column DAY. Order the results by the day of the week, starting with Monday.

SELECT last_name, hire_date, TO_CHAR(hire_date, 'Day') AS "DAY" FROM employees ORDER BY TO_CHAR(hire_date, 'D') ASC;

Practice Questions
Introduction to Functions

1. For each task, choose whether a single-row or multiple row function would be most appropriate: a. Showing all of the email addresses in upper case letters
b. Determining the average salary for the employees in the sales department c. Showing hire dates with the month spelled out *(September 1, 2004)*
d. Finding out the employees in each department that had the most seniority (the earliest hire date)
e. Displaying the employees' salaries rounded to the hundreds place
f. Substituting zeros for null values when displaying employee commissions.

a. Showing all of the email addresses in upper case letters: Single-row function
b. Determining the average salary for the employees in the sales department: Multiple-row function
c. Showing hire dates with the month spelled out (September 1, 2004): Single-row function
d. Finding out the employees in each department that had the most seniority (the earliest hire date): Multiple-row function
e. Displaying the employees' salaries rounded to the hundreds place: Single-row function
f. Substituting zeros for null values when displaying employee commissions: Single-row function

2.The most common multiple-row functions are: AVG, COUNT, MAX, MIN, and SUM. Give your own definition for each of these functions.

AVG: Calculates the average value of a numeric column.
COUNT: Counts the number of rows in a column or table.
MAX: Finds the maximum value in a column.
MIN: Finds the minimum value in a column.
SUM: Calculates the total sum of a numeric column.

3. Test your definitions by substituting each of the multiple-row functions in this query.
SELECT FUNCTION(salary) FROM employees
Write out each query and its results.

SELECT AVG(salary) FROM employees;
Result: Average salary of all employees.
SELECT COUNT(salary) FROM employees;
Result: Number of employees.
SELECT MAX(salary) FROM employees;
Result: Highest salary of all employees.
SELECT MIN(salary) FROM employees;
Result: Lowest salary of all employees.
SELECT SUM(salary) FROM employees;
Result: Total sum of all employees' salaries.

Case and Character Manipulation

1. Using the three separate words “Oracle,” “Internet,” and “Academy,” use one command to produce the following output:
The Best Class Oracle Internet Academy

```
SELECT 'The Best Class ' || 'Oracle' || ' ' || 'Internet' || ' ' || 'Academy' FROM DUAL;
```

2. Use the string “Oracle Internet Academy” to produce the following output: The Net net

```
SELECT 'The Net ' || SUBSTR('Oracle Internet Academy', 8, 3) || ' ' || LOWER(SUBSTR('Oracle Internet Academy', 8, 3)) FROM DUAL;
```

3. What is the length of the string “Oracle Internet Academy”?

```
SELECT LENGTH('Oracle Internet Academy') FROM DUAL;
```

4. What’s the position of “I” in “Oracle Internet Academy”?

```
SELECT INSTR('Oracle Internet Academy', 'I') FROM DUAL;
```

5. Starting with the string “Oracle Internet Academy”, pad the string to create
****Oracle****Internet****Academy****

```
SELECT RPAD('****' || 'Oracle', 11, '*') || RPAD('****' || 'Internet', 20, '*') || RPAD('****' || 'Academy', 30, '*') FROM DUAL;
```

Number Functions

1. Display Oracle database employee last_name and salary for employee_ids between 100 and 102. Include a third column that divides each salary by 1.55 and rounds the result to two decimal places.

SELECT last_name, salary, ROUND(salary / 1.55, 2) FROM employees WHERE employee_id BETWEEN 100 AND 102;

2. Display employee last_name and salary for those employees who work in department 80. Give each of them a raise of 5.333% and truncate the result to two decimal places.

SELECT last_name, TRUNC(salary * 1.05333, 2) AS salary FROM employees WHERE department_id = 80;

3. Use a MOD number function to determine whether 38873 is an even number or an odd number.

SELECT MOD(38873, 2) FROM DUAL;

4. Use the DUAL table to process the following numbers:
845.553 - round to one decimal place
30695.348 - round to two decimal places
30695.348 - round to -2 decimal Places
2.3454 - truncate the 454 from the decimal place

SELECT ROUND(845.553, 1) AS rounded_one, ROUND(30695.348, 2) AS rounded_two, ROUND(30695.348, -2) AS rounded_negative_two, TRUNC(2.3454, 2) AS truncated FROM DUAL;

5. Divide each employee's salary by 3. Display only those employees' last names and salaries who earn a salary that is a multiple of 3.

SELECT last_name, salary FROM employees WHERE MOD(salary, 3) = 0;

6. Divide 34 by 8. Show only the remainder of the division. Name the output as EXAMPLE.

SELECT MOD(34, 8) AS EXAMPLE FROM DUAL;

7. How would you like your paycheck – rounded or truncated? What if your paycheck was calculated to be $565.784 for the week, but you noticed that it was issued for $565.78. The loss of .004 cent would probably make very little difference to you. However, what if this was done to a thousand people, a 100,000 people, or a million people! Would it make a difference then? How much difference?

SELECT ROUND(565.784, 2) AS rounded_pay FROM DUAL;

SELECT TRUNC(565.784, 2) AS truncated_pay FROM DUAL;

SELECT 0.004 * 1000000 AS total_loss FROM DUAL;

EXERCISE-7

Displaying data from multiple tables

Find the Solution for the following:

1. Write a query to display the last name, department number, and department name for all employees.

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

2. Create a unique listing of all jobs that are in department 80. Include the location of the department in the output.

```
SELECT DISTINCT j.job_id, j.job_title, d.location_id FROM employees e
JOIN jobs j ON e.job_id = j.job_id
JOIN departments d ON e.department_id = d.department_id
WHERE e.department_id = 80;
```

3. Write a query to display the employee last name, department name, location ID, and city of all employees who earn a commission

```
SELECT e.last_name, d.department_name, d.location_id, l.city FROM employees e JOIN
 departments d ON e.department_id = d.department_id
JOIN locations l ON d.location_id = l.location_id WHERE e.commission_pct IS NOT NULL;
```

4. Display the employee last name and department name for all employees who have an a(lowercase) in their last names. P

```
SELECT e.last_name, d.department_name FROM employees e JOIN departments
 d ON e.department_id = d.department_id WHERE LOWER(e.last_name) LIKE '%a%';
```

5. Write a query to display the last name, job, department number, and department name for all employees who work in Toronto.

```
SELECT e.last_name, j.job_title, e.department_id, d.department_name FROM employees e
JOIN jobs j ON e.job_id = j.job_id JOIN departments d ON e.department_id = d.department_id
JOIN locations l ON d.location_id = l.location_id WHERE l.city = 'Toronto';
```

6. Display the employee last name and employee number along with their manager‘s last name and 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 LEFT JOIN employees m ON e.manager_id = m.employee_id;
```

7. Modify lab4_6.sql to display all employees including King, who has no manager. Order the results by the employee number.

```
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.employee_id;
```

8. Create a query that displays employee last names, department numbers, and all the employees who work in the same department as a given employee. Give each column an appropriate label

```
SELECT e1.last_name AS Employee, e1.department_id AS Dept#, e2.last_name AS Colleague
FROM employees e1
JOIN employees e2 ON e1.department_id = e2.department_id
ORDER BY e1.department_id, e1.last_name, e2.last_name;
```

9. Show the structure of the JOB_GRADES table. Create a query that displays the name, job, department name, salary, and grade for all employees

```
DESC JOB_GRADES;
SELECT e.last_name AS Name,
    e.job_id AS Job,
    d.department_name AS Department,
    e.salary AS Salary,
    j.grade_level AS Grade
FROM employees e
JOIN departments d ON e.department_id = d.department_id
JOIN job_grades j ON e.salary BETWEEN j.lowest_sal AND j.highest_sal;
```

10. Create a query to display the name and hire date of any employee hired after employee Davies.

```
SELECT e.last_name AS Name, e.hire_date AS Hire_Date FROM employees e
WHERE e.hire_date > (SELECT hire_date FROM employees WHERE last_name = 'Davies');
```

11. Display the names and hire dates for all employees who were hired before their managers, along with their manager‘s 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;
```

EXERCISE-8

Aggregating Data Using Group Functions

Find the Solution for the following:

Determine the validity of the following three statements. Circle either True or False.

1. Group functions work across many rows to produce one result per group. True/False

True

2. Group functions include nulls in calculations. True/False

False

3. The WHERE clause restricts rows prior to inclusion in a group calculation. True/False

True

The HR department needs the following reports:

4. Find the highest, lowest, sum, and average salary of all employees. Label the columns Maximum, Minimum, Sum, and Average, respectively. Round your results to the nearest whole number

```
SELECT ROUND(MAX(salary)) AS Maximum, ROUND(MIN(salary)) AS Minimum,

  ROUND(SUM(salary)) AS Sum, ROUND(AVG(salary)) AS Average FROM employees;
```

5. Modify the above query to display the minimum, maximum, sum, and average salary for each job type.

```
SELECT job_id,
      ROUND(MAX(salary)) AS Maximum,
      ROUND(MIN(salary)) AS Minimum,
      ROUND(SUM(salary)) AS Sum,
      ROUND(AVG(salary)) AS Average
FROM employees
GROUP BY job_id;
```

6.Write a query to display the number of people with the same job. Generalize the query so that the user in the HR department is prompted for a job title.

```
SELECT job_id, COUNT(*) AS Number_of_People
FROM employees
```

```
WHERE job_id = :job_title
GROUP BY job_id;
```

7. Determine the number of managers without listing them. Label the column Number of Managers.
Hint: Use the MANAGER_ID column to determine the number of managers.

```
SELECT COUNT(DISTINCT manager_id) AS Number_of_Managers
FROM employees
WHERE manager_id IS NOT NULL;
```

8. Find the difference between the highest and lowest salaries. Label the column DIFFERENCE.

```
SELECT MAX(salary) - MIN(salary) AS DIFFERENCE
FROM employees;
```

9. Create a report to display the manager number and the salary of the lowest-paid employee for that manager. Exclude anyone whose manager is not known. Exclude any groups where the minimum salary is $6,000 or less. Sort the output in descending order of salary.

```
SELECT  manager_id, MIN(salary) AS min_salary FROM employees WHERE manager_id IS
NOT NULL GROUP BY manager_id HAVING MIN(salary) > 6000 ORDER BY min_salary
DESC;
```

10. Create a query to display the total number of employees and, of that total, the number of employees hired in 1995, 1996, 1997, and 1998. Create appropriate column headings.

```
SELECT
  COUNT(*) AS total_employees,
  SUM(CASE WHEN EXTRACT(YEAR FROM hire_date) = 1995 THEN 1 ELSE 0
END) AS hired_1995,
  SUM(CASE WHEN EXTRACT(YEAR FROM hire_date) = 1996 THEN 1 ELSE 0
END) AS hired_1996,
  SUM(CASE WHEN EXTRACT(YEAR FROM hire_date) = 1997 THEN 1 ELSE 0
END) AS hired_1997,
  SUM(CASE WHEN EXTRACT(YEAR FROM hire_date) = 1998 THEN 1 ELSE 0
END) AS hired_1998
FROM
  Employees;
```

11. Create a matrix query to display the job, the salary for that job based on department number, and the total salary for that job, for departments 20, 50, 80, and 90, giving each column an appropriate heading.

```
SELECT job_id,SUM(CASE WHEN department_id = 20 THEN salary ELSE 0 END)
AS dept_20_salary,SUM(CASE WHEN department_id = 50 THEN salary ELSE 0
END) AS dept_50_salary,SUM(CASE WHEN department_id = 80 THEN salary ELSE
0 END) AS dept_80_salary,SUM(CASE WHEN department_id = 90 THEN salary
ELSE 0 END) AS dept_90_salary,SUM(salary) AS total_salary FROM employees
WHERE department_id IN (20, 50, 80, 90)GROUP BY job_id;
```

12.Write a query to display each department‘s name, location, number of employees, and the average salary for all the employees in that department. Label the column name-Location, Number of people, and salary respectively. Round the average salary to two decimal places.

```
SELECT d.department_name AS department_name, l.city AS location,
COUNT(e.employee_id) AS number_of_people,ROUND(AVG(e.salary), 2) AS
average_salary FROM departments d JOIN employees e ON d.department_id =
e.department_id JOIN locations l ON d.location_id = l.location_id
GROUP BY d.department_name, l.city;
```

Practice Questions

Date Functions

1. For DJs on Demand, display the number of months between the event_date of the Vigil wedding and today's date. Round to the nearest month.

SELECT ROUND(MONTHS_BETWEEN(SYSDATE, (SELECT event_date FROM events WHERE event_name = 'Vigil wedding'))) AS months_difference FROM dual;

2. Display the days between the start of last summer's school vacation break and the day school started this year. Assume 30.5 days per month. Name the output "Days."

SELECT ROUND((this_year_start - last_summer_start) * 30.5, 0) AS Days FROM dual;

3. Display the days between January 1 and December 31.

SELECT (TO_DATE('31-DEC', 'DD-MON') - TO_DATE('01-JAN', 'DD-MON')) AS days_difference

FROM dual;

4. Using one statement, round today's date to the nearest month and nearest year and truncate it to the nearest month and nearest year. Use an alias for each column.

SELECT ROUND(SYSDATE, 'MONTH') AS rounded_month, ROUND(SYSDATE, 'YEAR') AS rounded_year,

TRUNC(SYSDATE, 'MONTH') AS truncated_month,TRUNC(SYSDATE, 'YEAR') AS truncated_year FROM

Dual;

5. What is the last day of the month for June 2005? Use an alias for the output.

SELECT LAST_DAY(TO_DATE('01-JUN-2005', 'DD-MON-YYYY')) AS last_day_of_june_2005

FROM dual;

6.Display the number of years between the Global Fast Foods employee Bob Miller's birthday and today. Round to the nearest year.

SELECT ROUND(MONTHS_BETWEEN(SYSDATE, (SELECT birth_date FROM employees WHERE name =

'Bob Miller')) / 12, 0) AS years_difference

FROM dual;

7.Your next appointment with the dentist is six months from today. On what day will you go to the dentist? Name the output, "Appointment."

SELECT ADD_MONTHS(SYSDATE, 6) AS Appointment

FROM dual;

8.The teacher said you have until the last day of this month to turn in your research paper. What day will this be? Name the output, “Deadline.”

SELECT LAST_DAY(SYSDATE) AS Deadline FROM dual;

9.How many months between your birthday this year and January 1 next year? 10.What’s the date of the next Friday after your birthday this year? Name the output, “First Friday.”

SELECT MONTHS_BETWEEN(TO_DATE('01-JAN-YYYY', 'DD-MON-YYYY') + 1,

TO_DATE('your_birthday_this_year', 'DD-MON-YYYY')) AS months_difference

FROM dual;

11. Name a date function that will return a number.

MONTHS_BETWEEN(date1, date2)

12.Name a date function that will return a date.

ADD_MONTHS(date, n)

13.Give one example of why it is important for businesses to be able to manipulate date data?

It is crucial for businesses to manipulate date data for accurate scheduling, tracking deadlines, forecasting, and generating timely reports which are essential for decision-making and operational efficiency.

Conversion Functions

In each of the following exercises, feel free to use labels for the converted column to make the output more readable.

1. List the last names and birthdays of Global Fast Food Employees. Convert the birth dates to character data in the Month DD, YYYY format. Suppress any leading zeros.

```
SELECT last_name, TO_CHAR(birth_date, 'Month DD, YYYY') AS birth_date_formatted

FROM employees;
```

2. Convert January 3, 04, to the default date format 03-Jan-2004.

```
SELECT TO_DATE('03-Jan-2004', 'DD-MON-YYYY') AS converted_date FROM dual;
```

3. Format a query from the Global Fast Foods f_promotional_menus table to print out the start_date of promotional code 110 as: The promotion began on the tenth of February 2004.

```
SELECT 'The promotion began on the ' || TO_CHAR(start_date, 'DDth of Month YYYY') AS

promotion_start FROM f_promotional_menus WHERE promo_code = 110;
```

4. Convert today's date to a format such as: "Today is the Twentieth of March, Two Thousand Four"

```
SELECT 'Today is the ' || TO_CHAR(SYSDATE, 'DDth of Month, YYYY') AS formatted_today

FROM dual;
```

5. List the ID, name and salary for all Global Fast Foods employees. Display salary with a $ sign and two decimal places.

```
SELECT id, name, TO_CHAR(salary, '$999,999.99') AS formatted_salary

FROM employees;
```

EXERCISE-9

Sub queries

Find the Solution for the following:

1. The HR department needs a query that prompts the user for an employee last name. The query then displays the last name and hire date of any employee in the same department as the employee whose name they supply (excluding that employee). For example, if the user enters Zlotkey, find all employees who work with Zlotkey (excluding Zlotkey).

SELECT e.last_name, e.hire_date FROM employees e JOIN employees target ON e.department_id = target.department_id WHERE target.last_name = :employee_last_name AND e.employee_id != target.employee_id;

2. Create a report that displays the employee number, last name, and salary of all employees who earn more than the average salary. Sort the results in order of ascending salary.

SELECT employee_id, last_name, salary FROM employees WHERE salary > (SELECT AVG(salary) FROM employees) ORDER BY salary ASC;

3. Write a query that displays the employee number and last name of all employees who work in a department with any employee whose last name contains a *u*.

SELECT employee_id, last_name FROM employees WHERE department_id IN (SELECT department_id FROM employees WHERE last_name LIKE '%u%');

4. The HR department needs a report that displays the last name, department number, and job ID of all employees whose department location ID is 1700.

SELECT last_name, department_id, job_id FROM employees WHERE department_id IN (SELECT department_id FROM departments WHERE location_id = 1700);

5. Create a report for HR that displays the last name and salary of every employee who reports to King.

SELECT e.last_name, e.salary FROM employees e JOIN employees mgr ON e.manager_id = mgr.employee_id WHERE mgr.last_name = 'King';

6. Create a report for HR that displays the department number, last name, and job ID for every employee in the Executive department.

SELECT department_id, last_name, job_id FROM employees WHERE department_id = (SELECT department_id FROM departments WHERE department_name = 'Executive');

7. Modify the query 3 to display the employee number, last name, and salary of all employees who earn more than the average salary and who work in a department with any employee whose last name contains a *u*.

SELECT employee_id, last_name, salary FROM employees WHERE salary > (SELECT AVG(salary) FROM employees) AND department_id IN (SELECT department_id FROM employees WHERE last_name LIKE '%u%');

Practice Questions

1. Ellen Abel is an employee who has received a $2,000 raise. Display her first name and last name,her current salary, and her new salary. Display both salaries with a $ and two decimal places. Label her new salary column AS New Salary.

SELECT first_name, last_name, TO_CHAR(salary, '$9999.99') AS current_salary, TO_CHAR(salary + 2000, '$9999.99') AS "New Salary" FROM employees WHERE first_name = 'Ellen' AND last_name = 'Abel';

2. On what day of the week and date did Global Fast Foods' promotional code 110 Valentine's Special begin?

SELECT TO_CHAR(start_date, 'Day, DD-Mon-YYYY') AS start_day_date FROM promotions WHERE promo_code = 110;

3. Create one query that will convert 25-Dec-2004 into each of the following (you will have to convert 25-Dec-2004 to a date and then to character data):

December 25th, 2004

DECEMBER 25TH, 2004

25th december, 2004

SELECT TO_CHAR(TO_DATE('25-Dec-2004', 'DD-Mon-YYYY'), 'Month DDth, YYYY') AS "December 25th, 2004",TO_CHAR(TO_DATE('25-Dec-2004', 'DD-Mon-YYYY'), 'MONTH DDTH, YYYY') AS "DECEMBER 25TH, 2004",TO_CHAR(TO_DATE('25-Dec-2004', 'DD-Mon-YYYY'), 'DDth month, YYYY') AS "25th december, 2004"FROM dual;

4. Create a query that will format the DJs on Demand d_packages columns, low-range and high-range package costs, in the format $2500.00.

SELECT TO_CHAR(low_range, '$9999.99') AS low_range, TO_CHAR(high_range, '$9999.99') AS high_range FROM d_packages;

5. Convert JUNE192004 to a date using the fx format model.

SELECT TO_DATE('JUNE192004', 'fxMONTHDDYYYY') AS formatted_date FROM dual;

6. What is the distinction between implicit and explicit datatype conversion? Give an example of each.

Implicit Datatype Conversion:

Happens automatically by the database when needed.

Explicit Datatype Conversion:

Requires a function to explicitly convert the datatype.

7. Why is it important from a business perspective to have datatype conversions?

Datatype conversions are crucial for ensuring that data is accurately interpreted and processed in various contexts. Proper datatype conversion allows for:

Accurate Calculations: Ensuring numbers are treated as numbers for arithmetic operations.

Correct Comparisons: Enabling valid comparisons between data of different types.

Data Integrity: Preventing data corruption by enforcing the correct datatype.

System Interoperability: Facilitating data exchange between systems with different datatype requirements.

These factors support business operations by maintaining data accuracy, reliability, and consistency across applications and systems.

EXERCISE-10

Find the Solution for the following:

1. The HR department needs a list of department IDs for departments that do not contain the job ID ST_CLERK. Use set operators to create this report.

SELECT department_id FROM departments MINUS

SELECT department_id FROM employees WHERE job_id = 'ST_CLERK';

2. The HR department needs a list of countries that have no departments located in them.Display the country ID and the name of the countries. Use set operators to create this report.

SELECT country_id, country_name FROM countries MINUS SELECT c.country_id, c.country_name

FROM countries c JOIN locations l ON c.country_id = l.country_id JOIN departments d ON

l.location_id = d.location_id;

3. Produce a list of jobs for departments 10, 50, and 20, in that order. Display job ID and department ID using set operators.

(SELECT job_id, department_id FROM employees WHERE department_id = 10) UNION

(SELECT job_id, department_id FROM employees WHERE department_id = 50)UNION

(SELECT job_id, department_id FROM employees WHERE department_id = 20);

4. Create a report that lists the employee IDs and job IDs of those employees who currently have a job title that is the same as their job title when they were initially hired by the company (that is, they changed jobs but have now gone back to doing their original job).

SELECT employee_id, job_id FROM job_historyWHERE job_id = (SELECT job_id FROM

employees WHERE job_history.employee_id = employees.employee_id) AND start_date = (SELECT

MIN(start_date) FROM job_history WHERE job_history.employee_id = employees.employee_id);

5. The HR department needs a report with the following specifications:Last name and department ID of all the employees from the EMPLOYEES table, regardless of whether or not they belong to a department.- Department ID and department name of all the departments from the DEPARTMENTS table,regardless of whether or not they have employees working in them Write a compound query to accomplish this.

SELECT e.last_name, e.department_id FROM employees e UNION SELECT d.department_id,

d.department_name FROM departments d;

Practice Exercise

1. Create a report that shows the Global Fast Foods promotional name, start date, and end date from the f_promotional_menus table. If there is an end date, temporarily replace it with “end in two weeks”. If there is no end date, replace it with today’s date.

SELECT promotional_name, start_date, NVL(TO_CHAR(end_date, 'end in
two weeks'), TO_CHAR(SYSDATE, 'DD-Mon-YYYY')) AS end_date FROM f_promotional_menus;

2. Not all Global Fast Foods staff members receive overtime pay. Instead of displaying a null value for these employees, replace null with zero. Include the employee’s last name and overtime rate in the output. Label the overtime rate as “Overtime Status”.

SELECT last_name, NVL(overtime_rate, 0) as "Overtime Status"
FROM Global_Fast_Foods_Staff;

3. The manager of Global Fast Foods has decided to give all staff who currently do not earn overtime an overtime rate of $5.00. Construct a query that displays the last names and the overtime rate for each staff member, substituting $5.00 for each null overtime value.

SELECT last_name, NVL(overtime_rate, 5.00) as "Overtime Rate"
FROM Global_Fast_Foods_Staff;

4. Not all Global Fast Foods staff members have a manager. Create a query that displays the employee last name and 9999 in the manager ID column for these employees.

SELECT last_name, NVL(manager_id, 9999) as "Manager ID"
FROM Global_Fast_Foods_Staff;

5. Which statement(s) below will return null if the value of v_sal is 50?
a. SELECT nvl(v_sal, 50) FROM emp;
b. SELECT nvl2(v_sal, 50) FROM emp;
c. SELECT nullif(v_sal, 50) FROM emp;
d. SELECT coalesce (v_sal, Null, 50) FROM emp;

Statement c will return null if v_sal is 50. It uses the NULLIF function.

6. What does this query on the Global Fast Foods table return?

SELECT COALESCE(last_name, to_char(manager_id)) as NAME
FROM f_staffs;

7a. Create a report listing the first and last names and month of hire for all employees in the EMPLOYEES table (use TO_CHAR to convert hire_date to display the month).

```
SELECT first_name, last_name, TO_CHAR(hire_date, 'Month') as month_of_hire
FROM employees;
```

b.Modify the report to display null if the month of hire is September. Use the NULLIF function.

```
SELECT
first_name, last_name, NULLIF(TO_CHAR(hire_date, 'Month'), 'September') as month_of_hire
FROM employees;
```

8. For all null values in the specialty column in the DJs on Demand d_partners table, substitute “No Specialty.” Show the first name and specialty columns only.

Conditional Expressions

1. From the DJs on Demand d_songs table, create a query that replaces the 2-minute songs with “shortest” and the 10-minute songs with “longest”. Label the output column “Play Times”.

```
SELECT
  CASE
    WHEN duration = 2 THEN 'Shortest'
    WHEN duration = 10 THEN 'Longest'
    ELSE duration
  END as "Play Times"
FROM d_songs;
```

2. Use the Oracle database employees table and CASE expression to decode the department id.
on Display the department id, last name, salary and a column called “New Salary” whose value is based the following conditions:
If the department id is 10 then 1.25 * salary If
the department id is 90 then 1.5 * salary
If the department id is 130 then 1.75 * salary
Otherwise, display the old salary.

```
SELECT department_id, last_name, salary,
  CASE
    WHEN department_id = 10 THEN 1.25 * salary
    WHEN department_id = 90 THEN 1.5 * salary
    WHEN department_id = 130 THEN 1.75 * salary
    ELSE salary
  END as "New Salary"
FROM employees;
```

3. Display the first name, last name, manager ID, and commission percentage of all employees in departments 80 and 90. In a 5th column called “Review”, again display the manager ID. If they don’t have a manager, display the commission percentage. If they don’t have a commission, display 99999.

```
SELECT first_name, last_name, manager_id, commission_pct,
  CASE
```

```
    WHEN manager_id IS NULL THEN commission_pct
    WHEN commission_pct IS NULL THEN 99999
    ELSE manager_id
  END as "Review"
FROM employees
WHERE department_id IN (80, 90);
```

Cross Joins and Natural Joins

Use the Oracle database for problems 1-4.
1. Create a cross-join that displays the last name and department name from the employees and departments tables.

```
SELECT e.last_name, d.department_name
FROM employees e
CROSS JOIN departments d;
```

2. Create a query that uses a natural join to join the departments table and the locations table. Display the department id, department name, location id, and city.

```
SELECT d.department_id, d.department_name, l.location_id, l.city
FROM departments d
NATURAL JOIN locations l;
```

3. Create a query that uses a natural join to join the departments table and the locations table. Restrict the output to only department IDs of 20 and 50. Display the department id, department name, location id, and city.

```
SELECT d.department_id, d.department_name, l.location_id, l.city FROM departments d
NATURAL JOIN locations l
WHERE d.department_id IN (20, 50);
```

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';
```

EXERCISE 13

1. What is a “constraint” as it relates to data integrity?

A constraint in the context of databases is a rule applied to columns in a table to ensure the accuracy and reliability of the data within the table Constraints enforce data integrity by limiting the type of data that can be inserted into a column, thus preventing invalid data from entering the database.

2. What are the limitations of constraints that may be applied at the column level and at the table Level?

Constraints like NOT NULL, UNIQUE, PRIMARY KEY, and CHECK can be defined directly on a column.
Limited to the specific column only.

3. Why is it important to give meaningful names to constraints?

Enhance readability and maintainability of the database schema.
Help in easily identifying and understanding the purpose of the constraint.
Simplify troubleshooting and debugging when constraints are violated.

4. Based on the information provided by the owners, choose a datatype for each column. Indicate the length, precision, and scale for each NUMBER datatype.

Column Name	Data Type	Nullable
location_id	NUMBER(5, 0)	NO
street_address	VARCHAR2(50)	YES
postal_code	VARCHAR2(12)	YES
city	VARCHAR2(30)	NO
state_province	VARCHAR2(25)	YES
country_id	CHAR(2)	YES

5. Use “(nullable)” to indicate those columns that can have null values.

Column Name	Data Type	Nullable
location_id	NUMBER(5, 0)	NO
street_address	VARCHAR2(50)	YES
postal_code	VARCHAR2(12)	YES
city	VARCHAR2(30)	NO
state_province	VARCHAR2(25)	YES
country_id	CHAR(2)	YES

6. Write the CREATE TABLE statement for the Global Fast Foods locations table to define the constraints at the column level.

CREATE TABLE locations (location_id NUMBER(5, 0) PRIMARY KEY,
street_address VARCHAR2(50),postal_code VARCHAR2(12)city VARCHAR2(30) NOT NULL,
state_province VARCHAR2(25), country_id CHAR(2));

7. Execute the CREATE TABLE statement in Oracle Application Express.

Executing the create table statem

8. Execute a DESCRIBE command to view the Table Summary information.

DESC locations;

9. Rewrite the CREATE TABLE statement for the Global Fast Foods locations table to define the UNIQUE constraints at the table level. Do not execute this statement.

CREATE TABLE locations (location_id NUMBER(5, 0),street_address VARCHAR2(50),
postal_code VARCHAR2(12),city VARCHAR2(30) NOT NULL,state_province VARCHAR2(25),
country_id CHAR(2), CONSTRAINT loc_pk PRIMARY KEY (location_id),
CONSTRAINT loc_city_uk UNIQUE (city));

PRIMARY KEY, FOREIGN KEY, and CHECK Constraints

1. What is the purpose of a
• PRIMARY KEY
• FOREIGN KEY
• CHECK CONSTRAINT

PRIMARY KEY: Ensures that each row in the table has a unique identifier and no NULL values.
FOREIGN KEY: Enforces a link between two tables, ensuring that the foreign key in the child table matches a primary key in the parent table.
CHECK CONSTRAINT: Ensures that all values in a column meet a specific condition.

2. Using the column information for the animals table below, name constraints where applicable at the table level, otherwise name them at the column level. Define the primary key (animal_id). The license_tag_number must be unique. The admit_date and vaccination_date columns cannot contain null values.
animal_id NUMBER(6)
name VARCHAR2(25)
license_tag_number NUMBER(10)
admit_date DATE
adoption_id NUMBER(5),
vaccination_date DATE

```
CREATE TABLE animals ( animal_id NUMBER(6) PRIMARY KEY, name VARCHAR2(25),
license_tag_number NUMBER(10) UNIQUE,admit_date DATE NOT NULL,
adoption_id NUMBER(5),vaccination_date DATE NOT NULL);
```

3. Create the animals table. Write the syntax you will use to create the table.

```
CREATE TABLE animals (animal_id NUMBER(6) CONSTRAINT animal_pk PRIMARY KEY,
name VARCHAR2(25),license_tag_number NUMBER(10) CONSTRAINT license_tag_uk UNIQUE,
admit_date DATE CONSTRAINT admit_date_nn NOT NULL,
adoption_id NUMBER(5),vaccination_date DATE CONSTRAINT vaccination_date_nn NOT NULL);
```

4. Enter one row into the table. Execute a SELECT * statement to verify your input. Refer to the graphic below for input.
ANIMAL_I D NAM E LICENSE_TAG_NUMBE R ADMIT_DAT E ADOPTION_I D VACCINATION_DAT E
101 Spot 35540 10-Oct-2004 205 12-Oct-2004

```
INSERT INTO animals (animal_id, name, license_tag_number, admit_date,
 adoption_id, vaccination_date)VALUES (101, 'Spot', 35540,
TO_DATE('10-OCT-2004', 'DD-MON-YYYY'), 205,
 TO_DATE('12-OCT-2004', 'DD-MON-YYYY'));

SELECT * FROM animals;
```

5. Write the syntax to create a foreign key (adoption_id) in the animals table that has a corresponding primary-key reference in the adoptions table. Show both the column-level and table-level syntax. Note that because you have not actually created an adoptions table, no adoption_id primary key exists, so the foreign key cannot be added to the animals table.

```
ALTER TABLE animals ADD CONSTRAINT animal_fk
 FOREIGN KEY (adoption_id) REFERENCES adoptions(adoption_id);
```

6. What is the effect of setting the foreign key in the ANIMAL table as:
a. ON DELETE CASCADE
b. ON DELETE SET NULL

ON DELETE CASCADE: Automatically deletes child records when the parent record is deleted.
ON DELETE SET NULL: Sets the foreign key to NULL in the child records when the parent record is deleted.

7. What are the restrictions on defining a CHECK constraint?

CHECK constraints must reference columns in the same table.
Cannot reference columns in other tables or subqueries.
Must evaluate to TRUE or FALSE for each row.

PRACTICE PROBLEM

Managing Constraints

1. What are four functions that an ALTER statement can perform on constraints?

Add a new constraint.
Drop an existing constraint.
Enable a disabled constraint.
Disable an active constraint.

2. Since the tables are copies of the original tables, the integrity rules are not passed onto the new tables; only the column datatype definitions remain. You will need to add a PRIMARY KEY constraint to the copy_d_clients table.
Name the primary key copy_d_clients_pk . What is the syntax you used to create the PRIMARY KEY constraint to the copy_d_clients.table?

```
ALTER TABLE copy_d_clients
ADD CONSTRAINT copy_d_clients_pk PRIMARY KEY (client_number);
```

3. Create a FOREIGN KEY constraint in the copy_d_events table. Name the foreign key copy_d_events_fk. This key references the copy_d_clients table client_number column. What is the syntax you used to create the FOREIGN KEY constraint in the copy_d_events table?

```
ALTER TABLE copy_d_events ADD CONSTRAINT copy_d_events_fk
FOREIGN KEY (client_number) REFERENCES copy_d_clients(client_number);
```

4. Use a SELECT statement to verify the constraint names for each of the tables. Note that the Table names must be capitalized.
a. The constraint name for the primary key in the copy_d_clients table is _.

```
SELECT CONSTRAINT_NAME FROM USER_CONSTRAINTS
WHERE TABLE_NAME = 'COPY_D_CLIENTS';
SELECT CONSTRAINT_NAME FROM USER_CONSTRAINTS
WHERE TABLE_NAME = 'COPY_D_EVENTS';
```

5. Drop the PRIMARY KEY constraint on the copy_d_clients table. Explain your results.

```
ALTER TABLE copy_d_clients
DROP CONSTRAINT copy_d_clients_pk;
```

6. Add the following event to the copy_d_events table. Explain your results.
ID NAME EVENT_DATE DESCRIPTION COST VENUE_ID PACKAGE_CODE THEME_CODE CLIENT_NUMBER
140 Cline
Bas

Mitzvah
15-Jul-2004 Church and
Private Home
formal
4500 105 87 77 7125

INSERT INTO copy_d_events (ID, NAME, EVENT_DATE,
DESCRIPTION, COST, VENUE_ID, PACKAGE_CODE, THEME_CODE, CLIENT_NUMBER)
VALUES (140, 'Cline Bas Mitzvah',
TO_DATE('15-JUL-2004', 'DD-MON-YYYY'), 'Church and Private Home formal',
4500, 105, 87, 77, 7125);

7. Create an ALTER TABLE query to disable the primary key in the copy_d_clients table. Then add the values from #6 to the copy_d_events table. Explain your results.

ALTER TABLE copy_d_clients DISABLE CONSTRAINT copy_d_clients_pk;

-- Insert the new event
INSERT INTO copy_d_events (ID, NAME, EVENT_DATE, DESCRIPTION, COST,
VENUE_ID, PACKAGE_CODE, THEME_CODE, CLIENT_NUMBER)
VALUES (140, 'Cline Bas Mitzvah', TO_DATE('15-JUL-2004', 'DD-MON-YYYY'),
'Church and Private Home formal', 4500, 105, 87, 77, 7125);

8. Repeat question 6: Insert the new values in the copy_d_events table. Explain your results.

ALTER TABLE copy_d_clients ENABLE CONSTRAINT copy_d_clients_pk;

9. Enable the primary-key constraint in the copy_d_clients table. Explain your results.

To re-enable referential integrity, ensure the data adheres to the constraint rules
before enabling it.

10. If you wanted to enable the foreign-key column and reestablish the referential integrity between these two tables, what must be done?

Disabling constraints allows data manipulation without constraint checks.
Re-enabling constraints ensures data integrity once the data manipulation is complete.

11. Why might you want to disable and then re-enable a constraint?

SELECT CONSTRAINT_NAME, CONSTRAINT_TYPE,
TABLE_NAME FROM USER_CONSTRAINTS;

12. Query the data dictionary for some of the constraints that you have created. How does the data dictionary identify each constraint type?

P: Primary key
R: Referential integrity (foreign key)
C: Check constraint
U: Unique constraint

EXERCISE 13

Creating Views

1. What are three uses for a view from a DBA's perspective?

Security: Restricting access to specific columns or rows within a table.
Simplification: Simplifying complex queries by encapsulating them within a view.
Data Aggregation: Providing aggregated data, such as summaries or statistics,
without exposing the raw data.

2. Create a simple view called view_d_songs that contains the ID, title and artist from the DJs on Demand table for each "New Age" type code. In the subquery, use the alias "Song Title" for the title Column.

```
CREATE VIEW view_d_songs AS
SELECT ID, title AS "Song Title", artist
FROM d_songs
WHERE type_code = 'New Age';
```

3. SELECT * FROM view_d_songs. What was returned?

```
SELECT * FROM view_d_songs;
```

4. REPLACE view_d_songs. Add type_code to the column list. Use aliases for all columns.
Or use alias after the CREATE statement as shown.

```
CREATE OR REPLACE VIEW view_d_songs AS
SELECT ID AS "Song ID", title AS "Song Title", artist AS "Artist Name",
type_code AS "Type Code"
FROM d_songs
WHERE type_code = 'New Age';
```

5. Jason Tsang, the disk jockey for DJs on Demand, needs a list of the past events and those planned for the coming months so he can make arrangements for each event's equipment setup. As the company manager, you do not want him to have access to the price that clients paid for their events. Create a view for Jason to use that displays the name of the event, the event date, and the theme description. Use aliases for each column name.

```
CREATE VIEW view_events_for_jason AS
SELECT name AS "Event Name", event_date AS "Event Date", theme_description AS "Theme"
FROM d_events;
```

6. It is company policy that only upper-level management be allowed access to individual employee salaries. The department managers, however, need to know the minimum, maximum, and average salaries, grouped by department. Use the Oracle database to prepare a view that displays the needed information for department managers.

```
CREATE VIEW view_dept_salaries AS
SELECT department_id,
      MIN(salary) AS "Min Salary",
      MAX(salary) AS "Max Salary",
      AVG(salary) AS "Avg Salary"
FROM employees
GROUP BY department_id;
```

DML Operations and Views

Use the DESCRIBE statement to verify that you have tables named copy_d_songs, copy_d_events, copy_d_cds, and copy_d_clients in your schema. If you don't, write a query to create a copy of each.
1. Query the data dictionary USER_UPDATABLE_COLUMNS to make sure the columns in the base tables will allow UPDATE, INSERT, or DELETE. All table names in the data dictionary are stored in Uppercase.
Use the same syntax but change table_name of the other tables.

```
DESC copy_d_songs;
DESC copy_d_events;
DESC copy_d_cds;
DESC copy_d_clients;
```

2. Use the CREATE or REPLACE option to create a view of all the columns in the copy_d_songs table called view_copy_d_songs.

```
SELECT * FROM USER_UPDATABLE_COLUMNS
WHERE table_name = 'COPY_D_SONGS';
```

3. Use view_copy_d_songs to INSERT the following data into the underlying copy_d_songs table. Execute a SELECT * from copy_d_songs to verify your DML command. See the graphic.
ID TITLE DURATION ARTIST TYPE_CODE
88 Mello Jello 2 The What 4

```
CREATE OR REPLACE VIEW view_copy_d_songs AS
SELECT * FROM copy_d_songs;
```

4. Create a view based on the DJs on Demand COPY_D_CDS table. Name the view read_copy_d_cds. Select all columns to be included in the view. Add a WHERE clause to restrict the year to 2000. Add the WITH READ ONLY option.

```
INSERT INTO view_copy_d_songs (ID, TITLE, DURATION, ARTIST, TYPE_CODE)
VALUES (88, 'Mello Jello', 2, 'The What', 4);
SELECT * FROM copy_d_songs;
```

5. Using the read_copy_d_cds view, execute a DELETE FROM read_copy_d_cds WHERE cd_number= 90;

```
CREATE VIEW read_copy_d_cds AS
SELECT * FROM copy_d_cds
WHERE year = 2000
WITH READ ONLY;
```

6. Use REPLACE to modify read_copy_d_cds. Replace the READ ONLY option with WITH CHECK OPTION CONSTRAINT ck_read_copy_d_cds. Execute a SELECT * statement to verify

that the view exists.

```
DELETE FROM read_copy_d_cds WHERE cd_number = 90;
-- This will fail due to the READ ONLY option.
```

7. Use the read_copy_d_cds view to delete any CD of year 2000 from the underlying copy_d_cds.

```
CREATE OR REPLACE VIEW read_copy_d_cds AS
SELECT * FROM copy_d_cds
WHERE year = 2000
WITH CHECK OPTION CONSTRAINT ck_read_copy_d_cds;
SELECT * FROM read_copy_d_cds;
```

8. Use the read_copy_d_cds view to delete cd_number 90 from the underlying copy_d_cds table.

```
DELETE FROM read_copy_d_cds WHERE year = 2000;
```

9. Use the read_copy_d_cds view to delete year 2001 records.

```
DELETE FROM read_copy_d_cds WHERE cd_number = 90;
```

10. Execute a SELECT * statement for the base table copy_d_cds. What rows were deleted?

```
SELECT * FROM copy_d_cds;
```

11.What are the restrictions on modifying data through a view?

Cannot modify data through a view that includes GROUP BY, DISTINCT, JOIN,
or aggregate functions.Views with READ ONLY or CHECK OPTION constraints
restrict DML operations.

12. What is Moore’s Law? Do you consider that it will continue to apply indefinitely? Support your opinion with research from the internet.

Moore's Law is the observation made by Gordon Moore, co-founder of Intel, in 1965, which states that the number of transistors on a microchip doubles approximately every two years,
though the cost of computers is halved.

13. What is the “singularity” in terms of computing?

The singularity, in the context of computing and artificial intelligence, refers to a hypothetical future point where technological growth becomes uncontrollable and irreversible, resulting in unfathomable changes to human civilization.

Managing Views

1. Create a view from the copy_d_songs table called view_copy_d_songs that includes only the title and artist. Execute a SELECT * statement to verify that the view exists.

```
CREATE VIEW view_copy_d_songs AS
SELECT title, artist
FROM copy_d_songs;
SELECT * FROM view_copy_d_songs;
```

2. Issue a DROP view_copy_d_songs. Execute a SELECT * statement to verify that the view has been Deleted.

```
DROP VIEW view_copy_d_songs;
SELECT * FROM view_copy_d_songs;
-- This will return an error as the view no longer exists.
```

3. Create a query that selects the last name and salary from the Oracle database. Rank the salaries from highest to lowest for the top three employees.

```
SELECT last_name, salary
FROM (SELECT last_name, salary, RANK() OVER (ORDER BY salary DESC) AS rnk
    FROM employees)
WHERE rnk <= 3;
```

4. Construct an inline view from the Oracle database that lists the last name, salary, department ID, and maximum salary for each department. Hint: One query will need to calculate maximum salary by department ID.

```
SELECT e.last_name, e.salary, e.department_id, d.max_salary
FROM employees e
JOIN (SELECT department_id, MAX(salary) AS max_salary
    FROM employees
    GROUP BY department_id) d
ON e.department_id = d.department_id;
```

5. Create a query that will return the staff members of Global Fast Foods ranked by salary from lowest to highest.

```
SELECT last_name, salary
FROM employees
ORDER BY salary ASC;
```

Indexes and Synonyms

1. What is an index and what is it used for?

An index improves data retrieval speed by providing quick access to rows based on the indexed columns.

2. What is a ROWID, and how is it used?

ROWID is a unique identifier for each row's physical location in the database.

3. When will an index be created automatically?

Indexes are automatically created for PRIMARY KEY and UNIQUE constraints.

4. Create a nonunique index (foreign key) for the DJs on Demand column (cd_number) in the D_TRACK_LISTINGS table. Use the Oracle Application Express SQL Workshop Data Browser to confirm that the index was created.

```
CREATE INDEX idx_cd_number ON d_track_listings (cd_number);
```

5. Use the join statement to display the indexes and uniqueness that exist in the data dictionary for the DJs on Demand D_SONGS table.

```
SELECT index_name, uniqueness
FROM USER_INDEXES
WHERE table_name = 'D_SONGS';
```

6. Use a SELECT statement to display the index_name, table_name, and uniqueness from the data dictionary USER_INDEXES for the DJs on Demand D_EVENTS table.

```
SELECT index_name, table_name, uniqueness
FROM USER_INDEXES
WHERE table_name = 'D_EVENTS';
```

7. Write a query to create a synonym called dj_tracks for the DJs on Demand d_track_listings table.

```
CREATE SYNONYM dj_tracks FOR d_track_listings;
```

8. Create a function-based index for the last_name column in DJs on Demand D_PARTNERS table that makes it possible not to have to capitalize the table name for searches. Write a SELECT statement that would use this index.

```
CREATE INDEX idx_last_name_lower
ON d_partners (LOWER(last_name));
SELECT * FROM d_partners WHERE LOWER(last_name) = 'smith';
```

9. Create a synonym for the D_TRACK_LISTINGS table. Confirm that it has been created by querying the data dictionary.

```
SELECT synonym_name
FROM USER_SYNONYMS
WHERE synonym_name = 'DJ_TRACKS';
```

10.Drop the synonym that you created in question

```
DROP SYNONYM dj_tracks;
```

EXERCISE 14

1. Create a sequence to be used with the primary key column of the DEPT table. The sequence should start at 200 and have a maximum value of 1000. Have your sequence increment by ten numbers. Name the sequence DEPT_ID_SEQ.

```
CREATE SEQUENCE DEPT_ID_SEQ
START WITH 200
INCREMENT BY 10
MAXVALUE 1000
NOCACHE
NOCYCLE;
```

2. Write a query in a script to display the following information about your sequences: sequence name, maximum value, increment size, and last number

```
SELECT sequence_name, max_value, increment_by, last_number
FROM user_sequences
WHERE sequence_name = 'DEPT_ID_SEQ';
```

3. Write a script to insert two rows into the DEPT table. Name your script lab12_3.sql. Be sure to use the sequence that you created for the ID column. Add two departments named Education and Administration. Confirm your additions. Run the commands in your script.

```
INSERT INTO DEPT (DEPT_ID, DEPT_NAME)
VALUES (DEPT_ID_SEQ.NEXTVAL, 'Education');
INSERT INTO DEPT (DEPT_ID, DEPT_NAME)
VALUES (DEPT_ID_SEQ.NEXTVAL, 'Administration');
SELECT * FROM DEPT WHERE DEPT_NAME IN ('Education', 'Administration');
```

4. Create a non unique index on the foreign key column (DEPT_ID) in the EMP table.

```
CREATE INDEX EMP_DEPT_ID_IDX
ON EMP (DEPT_ID);
```

5. Display the indexes and uniqueness that exist in the data dictionary for the EMP table.

```
SELECT index_name, uniqueness
FROM user_indexes
WHERE table_name = 'EMP';
```

EXERCISE 15

1. What privilege should a user be given to log on to the Oracle Server? Is this a system or an object privilege?

A user should be given the CREATE SESSION privilege to log on to the Oracle Server.
This is a system privilege.

2. What privilege should a user be given to create tables?

A user should be given the CREATE TABLE privilege to create tables. This is a system privilege.

3. If you create a table, who can pass along privileges to other users on your table?

The owner of the table (the user who created the table) can pass along privileges to other users on their table. Additionally, a user who has been granted the GRANT option on that table can also pass along privileges.

4. You are the DBA. You are creating many users who require the same system privileges. What should you use to make your job easier?

Create a role with the necessary system privileges and then grant that role to the users.

5. What command do you use to change your password?

ALTER USER username IDENTIFIED BY new_password;

6. Grant another user access to your DEPARTMENTS table. Have the user grant you query access to his or her DEPARTMENTS table.

```
-- User1 grants SELECT privilege to User2
GRANT SELECT ON DEPARTMENTS TO user2;
-- User2 grants SELECT privilege to User1
GRANT SELECT ON DEPARTMENTS TO user1;
```

7. Query all the rows in your DEPARTMENTS table.

```
SELECT * FROM DEPARTMENTS;
```

8. Add a new row to your DEPARTMENTS table. Team 1 should add Education as department number 500. Team 2 should add Human Resources department number 510. Query the other team‘s table.

```
INSERT INTO DEPARTMENTS (DEPARTMENT_ID, DEPARTMENT_NAME)
VALUES (500, 'Education');
INSERT INTO DEPARTMENTS (DEPARTMENT_ID, DEPARTMENT_NAME)
VALUES (510, 'Human Resources');
SELECT * FROM user1.DEPARTMENTS; -- Team 2 querying Team 1's table
SELECT * FROM user2.DEPARTMENTS; -- Team 1 querying Team 2's table
```

9. Query the USER_TABLES data dictionary to see information about the tables that you own.

```
SELECT * FROM USER_TABLES;
```

10.Revoke the SELECT privilege on your table from the other team.

```
REVOKE SELECT ON DEPARTMENTS FROM user2; -- Team 1 revoking access from Team 2
REVOKE SELECT ON DEPARTMENTS FROM user1; -- Team 2 revoking access from Team 1
```

11. Remove the row you inserted into the DEPARTMENTS table in step 8 and save the changes.

```
DELETE FROM DEPARTMENTS WHERE DEPARTMENT_ID = 500;
COMMIT;

DELETE FROM DEPARTMENTS WHERE DEPARTMENT_ID = 510;
COMMIT;
```

EXERCISE 16

PROGRAM 1

Write a PL/SQL block to calculate the incentive of an employee whose ID is 110.

```
DECLARE
  v_employee_id NUMBER := 110;
  v_salary employees.salary%TYPE;
  v_incentive NUMBER;
BEGIN
  SELECT salary INTO v_salary
  FROM employees
  WHERE employee_id = v_employee_id;

  -- Assuming incentive is 10% of the salary
  v_incentive := v_salary * 0.10;

  DBMS_OUTPUT.PUT_LINE('Incentive for employee ' || v_employee_id || ' is: ' ||
v_incentive);
EXCEPTION
  WHEN NO_DATA_FOUND THEN
    DBMS_OUTPUT.PUT_LINE('Employee not found.');
END;
```

PROGRAM 2

Write a PL/SQL block to show an invalid case-insensitive reference to a quoted and without quoted user-defined identifier.

```
DECLARE
  "CaseSensitiveVar" VARCHAR2(30) := 'Hello';
  caseSensitiveVar VARCHAR2(30);
BEGIN
  -- This will raise an error because PL/SQL is case-insensitive and won't differentiate between the identifiers.
  caseSensitiveVar := "CaseSensitiveVar";
  DBMS_OUTPUT.PUT_LINE(caseSensitiveVar);
END;
```

PROGRAM 3

Write a PL/SQL block to adjust the salary of the employee whose ID 122.

Sample table: employees

```
DECLARE
  v_employee_id NUMBER := 122;
  v_new_salary NUMBER := 5000;  -- Example of new salary
BEGIN
  UPDATE employees
  SET salary = v_new_salary
  WHERE employee_id = v_employee_id;

  IF SQL%ROWCOUNT > 0 THEN
    DBMS_OUTPUT.PUT_LINE('Salary updated for employee ' || v_employee_id);
  ELSE
    DBMS_OUTPUT.PUT_LINE('Employee not found.');
  END IF;
END;
```

PROGRAM 4

Write a PL/SQL block to create a procedure using the "IS [NOT] NULL Operator"

and show AND operator returns TRUE if and only if both operands are TRUE.

```
CREATE OR REPLACE PROCEDURE check_null_and_operator (
  p_input1 IN VARCHAR2,
  p_input2 IN VARCHAR2
) IS
BEGIN
  IF p_input1 IS NOT NULL AND p_input2 IS NOT NULL THEN
    DBMS_OUTPUT.PUT_LINE('Both inputs are not null.');
  ELSE
    DBMS_OUTPUT.PUT_LINE('One or both inputs are null.');
  END IF;
END;
/

BEGIN
  check_null_and_operator('Hello', 'World');  -- Both are not null
  check_null_and_operator(NULL, 'World');     -- One is null
END;
```

PROGRAM 5

Write a PL/SQL block to describe the usage of LIKE operator including wildcard characters and escape character.

```
DECLARE
  v_name VARCHAR2(50);
BEGIN
  FOR rec IN (SELECT first_name FROM employees WHERE
first_name LIKE 'S%' ESCAPE '\') LOOP
    DBMS_OUTPUT.PUT_LINE('Name: ' || rec.first_name);
  END LOOP;
END;
```

PROGRAM 6

Write a PL/SQL program to arrange the number of two variable in such a way that the small number will store in num_small variable and large number will store in num_large variable.

```
DECLARE
  num1 NUMBER := 10;
  num2 NUMBER := 20;
  num_small NUMBER;
  num_large NUMBER;
BEGIN
  IF num1 < num2 THEN
    num_small := num1;
    num_large := num2;
  ELSE
    num_small := num2;
    num_large := num1;
  END IF;
  DBMS_OUTPUT.PUT_LINE('Small number: ' || num_small || ', Large number: ' ||
num_large);
END;
```

PROGRAM 7

Write a PL/SQL procedure to calculate the incentive on a target achieved and

display the message either the record updated or not.

```
CREATE OR REPLACE PROCEDURE calculate_incentive (
  p_employee_id IN NUMBER,
  p_target_achieved IN NUMBER
) IS
  v_incentive NUMBER;
BEGIN
  v_incentive := p_target_achieved * 0.05;  -- Assuming 5% incentive
  UPDATE employees
  SET incentive = v_incentive
  WHERE employee_id = p_employee_id;
  IF SQL%ROWCOUNT > 0 THEN
    DBMS_OUTPUT.PUT_LINE('Incentive updated for employee ' || p_employee_id);
  ELSE
    DBMS_OUTPUT.PUT_LINE('Employee not found.');
  END IF;
END;/
BEGIN
  calculate_incentive(110, 10000);  -- Example call
END;
```

PROGRAM 8

Write a PL/SQL procedure to calculate incentive achieved according to the specific sale limit.

```
CREATE OR REPLACE PROCEDURE calculate_incentive_by_sales (
  p_employee_id IN NUMBER,
  p_sales IN NUMBER
) IS
  v_incentive NUMBER;
BEGIN
  IF p_sales > 10000 THEN
    v_incentive := p_sales * 0.10;  -- 10% for sales over 10000
  ELSE
    v_incentive := p_sales * 0.05;  -- 5% for sales 10000 or less
  END IF;
  UPDATE employees
  SET incentive = v_incentive
  WHERE employee_id = p_employee_id;
  IF SQL%ROWCOUNT > 0 THEN
    DBMS_OUTPUT.PUT_LINE('Incentive updated for employee ' || p_employee_id);
  ELSE
    DBMS_OUTPUT.PUT_LINE('Employee not found.');
  END IF;
END;/

BEGIN
  calculate_incentive_by_sales(110, 15000);  -- Example call
END;
```

PROGRAM 9

Write a PL/SQL program to count number of employees in department 50 and check whether this department have any vacancies or not. There are 45 vacancies in this department.

```
DECLARE
  v_dept_id NUMBER := 50;
  v_emp_count NUMBER;
  v_vacancies NUMBER := 45;
BEGIN
  SELECT COUNT(*) INTO v_emp_count
  FROM employees
  WHERE department_id = v_dept_id;

  IF v_emp_count < v_vacancies THEN
    DBMS_OUTPUT.PUT_LINE('Department 50 has ' || (v_vacancies - v_emp_count) || ' vacancies.');
  ELSE
    DBMS_OUTPUT.PUT_LINE('Department 50 has no vacancies.');
  END IF;
END;
```

PROGRAM 10

Write a PL/SQL program to count number of employees in a specific department and check whether this department have any vacancies or not. If any vacancies, how many vacancies are in that department.

```
DECLARE
  v_dept_id NUMBER := 50;  -- Example department ID
  v_emp_count NUMBER;
  v_vacancies NUMBER := 45;
BEGIN
  SELECT COUNT(*) INTO v_emp_count
  FROM employees
  WHERE department_id = v_dept_id;

  IF v_emp_count < v_vacancies THEN
    DBMS_OUTPUT.PUT_LINE('Department ' || v_dept_id || ' has ' || (v_vacancies - v_emp_count) || ' vacancies.');
  ELSE
    DBMS_OUTPUT.PUT_LINE('Department ' || v_dept_id || ' has no vacancies.');
  END IF;
END;
```

PROGRAM 11

Write a PL/SQL program to display the employee IDs, names, job titles, hire dates,

and salaries of all employees.

```
DECLARE
  CURSOR emp_cur IS
    SELECT employee_id, first_name || ' ' || last_name AS name, job_id, hire_date, salary
    FROM employees;
BEGIN
  FOR emp_rec IN emp_cur LOOP
    DBMS_OUTPUT.PUT_LINE('ID: ' || emp_rec.employee_id || ', Name: ' || emp_rec.name ||
                         ', Job Title: ' || emp_rec.job_id || ', Hire Date: ' || emp_rec.hire_date ||
                         ', Salary: ' || emp_rec.salary);
  END LOOP;
END;
```

PROGRAM 12

Write a PL/SQL program to display the employee IDs, names, and department

names of all employees.

```
DECLARE
  CURSOR emp_cur IS
    SELECT e.employee_id, e.first_name || ' ' || e.last_name AS name, d.department_name
    FROM employees e
    JOIN departments d ON e.department_id = d.department_id;
BEGIN
  FOR emp_rec IN emp_cur LOOP
    DBMS_OUTPUT.PUT_LINE('ID: ' || emp_rec.employee_id || ', Name: ' || emp_rec.name ||
                         ', Department: ' || emp_rec.department_name);
  END LOOP;
END;
```

PROGRAM 13
Write a PL/SQL program to display the job IDs, titles, and minimum salaries of all Jobs.

```
DECLARE
  CURSOR job_cur IS
    SELECT job_id, job_title, min_salary
    FROM jobs;
BEGIN
  FOR job_rec IN job_cur LOOP
    DBMS_OUTPUT.PUT_LINE('Job ID: ' || job_rec.job_id || ', Title: ' || job_rec.job_title ||
                         ', Min Salary: ' || job_rec.min_salary);
  END LOOP;
END;
```

PROGRAM 14
Write a PL/SQL program to display the employee IDs, names, and job history start dates of all employees.

```
DECLARE
  CURSOR emp_job_hist_cur IS
    SELECT e.employee_id, e.first_name || ' ' || e.last_name AS name, jh.start_date
    FROM employees e
    JOIN job_history jh ON e.employee_id = jh.employee_id;
BEGIN
  FOR rec IN emp_job_hist_cur LOOP
    DBMS_OUTPUT.PUT_LINE('Employee ID: ' || rec.employee_id || ', Name: ' || rec.name ||
                         ', Job History Start Date: ' || rec.start_date);
  END LOOP;
END;
```

PROGRAM 15
Write a PL/SQL program to display the employee IDs, names, and job history end dates of all employees.

```
DECLARE
  CURSOR emp_job_hist_cur IS
    SELECT e.employee_id, e.first_name || ' ' || e.last_name AS name, jh.end_date
    FROM employees e
    JOIN job_history jh ON e.employee_id = jh.employee_id;
BEGIN
  FOR rec IN emp_job_hist_cur LOOP
    DBMS_OUTPUT.PUT_LINE('Employee ID: ' || rec.employee_id || ', Name: ' || rec.name ||
                         ', Job History End Date: ' || rec.end_date);
  END LOOP;
END;
```

EXERCISE 17

Program 1

FACTORIAL OF A NUMBER USING FUNCTION

```
CREATE OR REPLACE FUNCTION calculate_factorial (
  p_number IN NUMBER
) RETURN NUMBER
IS
  v_result NUMBER := 1;
BEGIN
  FOR i IN 1..p_number LOOP
    v_result := v_result * i;
  END LOOP;
  RETURN v_result;
END;/
-- PL/SQL block to test the factorial function
DECLARE
  v_number NUMBER := 5;  -- Example number
  v_factorial NUMBER;
BEGIN
  v_factorial := calculate_factorial(v_number);
  DBMS_OUTPUT.PUT_LINE('Factorial of ' || v_number || ' is: ' || v_factorial);
END;
/
```

Program 2

Write a PL/SQL program using Procedures IN,INOUT,OUT parameters to retrieve the corresponding book information in library

```
CREATE OR REPLACE PROCEDURE get_book_info (
  p_book_id IN NUMBER,
  p_title OUT VARCHAR2,
  p_author OUT VARCHAR2,
  p_year_published IN OUT NUMBER
) IS
BEGIN
  SELECT title, author, year_published
  INTO p_title, p_author, p_year_published
  FROM books
  WHERE book_id = p_book_id;
  p_year_published := p_year_published + 1;
EXCEPTION
  WHEN NO_DATA_FOUND THEN
    DBMS_OUTPUT.PUT_LINE('Book not found.');
  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('An error occurred: ' || SQLERRM);
END;
/

DECLARE
  v_book_id NUMBER := 101;  -- Example book ID
  v_title VARCHAR2(100);
  v_author VARCHAR2(100);
  v_year_published NUMBER := 2000;
BEGIN
  get_book_info(v_book_id, v_title, v_author, v_year_published);
  DBMS_OUTPUT.PUT_LINE('Book ID: ' || v_book_id);
  DBMS_OUTPUT.PUT_LINE('Title: ' || v_title);
  DBMS_OUTPUT.PUT_LINE('Author: ' || v_author);
  DBMS_OUTPUT.PUT_LINE('Year Published (adjusted): ' || v_year_published);
END;
/
```

EXERCISE 18

Program 1

Write a code in PL/SQL to develop a trigger that enforces referential integrity by preventing the deletion of a parent record if child records exist.

```
CREATE TABLE parent_table (
  parent_id NUMBER PRIMARY KEY,
  parent_data VARCHAR2(100)
);
CREATE TABLE child_table (
  child_id NUMBER PRIMARY KEY,
  parent_id NUMBER,
  child_data VARCHAR2(100),
  FOREIGN KEY (parent_id) REFERENCES parent_table(parent_id)
);
CREATE OR REPLACE TRIGGER trg_prevent_parent_deletion
BEFORE DELETE ON parent_table
FOR EACH ROW
DECLARE
  v_child_count NUMBER;
BEGIN
  SELECT COUNT(*)
  INTO v_child_count
  FROM child_table
  WHERE parent_id = :OLD.parent_id;

  IF v_child_count > 0 THEN
    RAISE_APPLICATION_ERROR(-20001, 'Cannot delete parent record; child records
exist.');
  END IF;
END;/
```

Program 2

Write a code in PL/SQL to create a trigger that checks for duplicate values in a specific column and raises an exception if found.

```
CREATE TABLE unique_column_table (
  id NUMBER PRIMARY KEY,
  unique_data VARCHAR2(100)
);
CREATE OR REPLACE TRIGGER trg_check_duplicates
BEFORE INSERT OR UPDATE ON unique_column_table
FOR EACH ROW
DECLARE
  v_count NUMBER;
BEGIN
  SELECT COUNT(*)
  INTO v_count
  FROM unique_column_table
  WHERE unique_data = :NEW.unique_data
  AND id != :NEW.id;
  IF v_count > 0 THEN
    RAISE_APPLICATION_ERROR(-20002, 'Duplicate value found in unique_data column.');
  END IF;
END;
/
```

Program 3

Write a code in PL/SQL to create a trigger that restricts the insertion of new rows if the total of a column's values exceeds a certain threshold.

```
CREATE TABLE threshold_table (
  id NUMBER PRIMARY KEY,
  value NUMBER
);
CREATE OR REPLACE TRIGGER trg_restrict_inserts
BEFORE INSERT ON threshold_table
FOR EACH ROW
DECLARE
  v_total NUMBER;
  v_threshold CONSTANT NUMBER := 1000;  -- Set your threshold here
BEGIN
  SELECT SUM(value)
  INTO v_total
  FROM threshold_table;
  IF v_total + :NEW.value > v_threshold THEN
    RAISE_APPLICATION_ERROR(-20003, 'Total value exceeds the threshold.');
  END IF;
END;
/
```

Program 4

Write a code in PL/SQL to design a trigger that captures changes made to specific columns and logs them in an audit table.

```
CREATE TABLE original_table (
  id NUMBER PRIMARY KEY,
  sensitive_data VARCHAR2(100),
  other_data VARCHAR2(100)
);
CREATE TABLE audit_table (
  audit_id NUMBER PRIMARY KEY,
  id NUMBER,
  old_sensitive_data VARCHAR2(100),
  new_sensitive_data VARCHAR2(100),
  change_date DATE,
  user_name VARCHAR2(100)
);
CREATE OR REPLACE TRIGGER trg_capture_changes
AFTER UPDATE ON original_table
FOR EACH ROW
BEGIN
  IF :OLD.sensitive_data != :NEW.sensitive_data THEN
    INSERT INTO audit_table (audit_id, id, old_sensitive_data, new_sensitive_data,
 change_date, user_name)
    VALUES (audit_seq.NEXTVAL, :OLD.id, :OLD.sensitive_data, :NEW.sensitive_data,
SYSDATE, USER);
  END IF;
END;
/
```

Program 5

Write a code in PL/SQL to implement a trigger that records user activity (inserts,

updates, deletes) in an audit log for a given set of tables.

```
CREATE TABLE activity_log (
  log_id NUMBER PRIMARY KEY,
  table_name VARCHAR2(100),
  operation VARCHAR2(10),
  id NUMBER,
  activity_date DATE,
  user_name VARCHAR2(100)
);
CREATE OR REPLACE TRIGGER trg_log_activity
AFTER INSERT OR UPDATE OR DELETE ON original_table
FOR EACH ROW
BEGIN
  IF INSERTING THEN
    INSERT INTO activity_log (log_id, table_name, operation, id, activity_date, user_name)
    VALUES (activity_seq.NEXTVAL, 'original_table', 'INSERT', :NEW.id, SYSDATE,
USER);
  ELSIF UPDATING THEN
    INSERT INTO activity_log (log_id, table_name, operation, id, activity_date, user_name)
    VALUES (activity_seq.NEXTVAL, 'original_table', 'UPDATE', :NEW.id, SYSDATE,
 USER);
  ELSIF DELETING THEN
    INSERT INTO activity_log (log_id, table_name, operation, id, activity_date, user_name)
    VALUES (activity_seq.NEXTVAL, 'original_table', 'DELETE', :OLD.id, SYSDATE,
USER);
  END IF;
END;
/
```

Program 7

Write a code in PL/SQL to implement a trigger that automatically calculates and updates a running total column for a table whenever new rows are inserted.

```
CREATE TABLE sales (
  sale_id NUMBER PRIMARY KEY,
  sale_amount NUMBER,
  running_total NUMBER
);
CREATE OR REPLACE TRIGGER trg_update_running_total
AFTER INSERT ON sales
FOR EACH ROW
DECLARE
  v_running_total NUMBER;
BEGIN
  SELECT NVL(MAX(running_total), 0)
  INTO v_running_total
  FROM sales
  WHERE sale_id != :NEW.sale_id;
  UPDATE sales
  SET running_total = v_running_total + :NEW.sale_amount
  WHERE sale_id = :NEW.sale_id;
END;
/
```

Program 8

Write a code in PL/SQL to create a trigger that validates the availability of items before allowing an order to be placed, considering stock levels and pending orders.

```
CREATE TABLE items (
  item_id NUMBER PRIMARY KEY,
  item_name VARCHAR2(100),
  stock_level NUMBER
);
CREATE TABLE orders (
  order_id NUMBER PRIMARY KEY,
  item_id NUMBER,
  order_quantity NUMBER
);
CREATE OR REPLACE TRIGGER trg_validate_availability
BEFORE INSERT ON orders
FOR EACH ROW
DECLARE
  v_stock_level NUMBER;
BEGIN
  SELECT stock_level
  INTO v_stock_level
  FROM items
  WHERE item_id = :NEW.item_id;

  IF v_stock_level < :NEW.order_quantity THEN
    RAISE_APPLICATION_ERROR(-20004, 'Insufficient stock for item.');
  END IF;
END;
/
```

EXERCISE 19

Structure of 'restaurants' collection:


```
{
"address": {
"building": "1007",
"coord": [ -73.856077, 40.848447 ],
"street": "Morris Park Ave",
"zipcode": "10462"
},
"borough": "Bronx",
"cuisine": "Bakery",
"grades": [
{ "date": { "$date": 1393804800000 }, "grade": "A", "score": 2 },
{ "date": { "$date": 1378857600000 }, "grade": "A", "score": 6 },
{ "date": { "$date": 1358985600000 }, "grade": "A", "score": 10 },
{ "date": { "$date": 1322006400000 }, "grade": "A", "score": 9 },
{ "date": { "$date": 1299715200000 }, "grade": "B", "score": 14 }
],
"name": "Morris Park Bake Shop",
"restaurant_id": "30075445"
}
```


1. Write a MongoDB query to find the restaurant Id, name, borough and cuisine for those restaurants which prepared dish except 'American' and 'Chinees' or restaurant's name begins with letter 'Wil'.

```
db.restaurants.find(
  {
    $or: [
      { name: { $regex: /^Wil/ } },
      { cuisine: { $nin: ["American", "Chinees"] } }
    ]
  },
  { restaurant_id, name, borough, cuisine }
)
```

2. Write a MongoDB query to find the restaurant Id, name, and grades for those restaurants which achieved a grade of "A" and scored 11 on an ISODate "2014-08-11T00:00:00Z" among many of survey dates..

```
db.restaurants.find(
  {
   grades: {
     $elemMatch: {
       grade: "A",
       score: 11,
       date: ISODate("2014-08-11T00:00:00Z")
     }
   }
  },
  { restaurant_id, name, grades }
)
```

3. Write a MongoDB query to find the restaurant Id, name and grades for those restaurants where the 2nd element of grades array contains a grade of "A" and score 9 on an ISODate "2014-08-11T00:00:00Z".

```
db.restaurants.find(
  {
   $and: [
     { "grades.1.grade": "A" },
     { "grades.1.score": 9 },
     { "grades.1.date": ISODate("2014-08-11T00:00:00Z") }
   ]
  },
  { restaurant_id, name, grades }
)
```

5.Write a MongoDB query to find the restaurant Id, name, address and geographical location for those restaurants where 2nd element of coord array contains a value which is more than 42 and upto 52..

```
db.restaurants.find(
  {
   "address.coord.1": { $gt: 42, $lte: 52 }
  },
  { restaurant_id, name, address, "address.coord": 1 }
)
```

6.Write a MongoDB query to arrange the name of the restaurants in ascending order along with all the columns.

```
db.restaurants.find().sort({ name: 1 })
```

6. Write a MongoDB query to arrange the name of the restaurants in descending along with all the columns.

```
db.restaurants.find().sort({ name: -1 })
```

7.Write a MongoDB query to arranged the name of the cuisine in ascending order and for that same cuisine borough should be in descending order.

```
db.restaurants.find().sort({ cuisine: 1, borough: -1 })
```

8.Write a MongoDB query to know whether all the addresses contains the street or not.

```
db.restaurants.find({ "address.street": { $exists: true } })
```

9.Write a MongoDB query which will select all documents in the restaurants collection where the coord field value is Double.

```
db.restaurants.find(
  { "address.coord": { $type: "double" } }
)
```

10. Write a MongoDB query which will select the restaurant Id, name and grades for those restaurants which returns 0 as a remainder after dividing the score by 7.

```
db.restaurants.find(
  {
    grades: {
      $elemMatch: {
        score: { $mod: [7, 0] }
      }
    }
  },
  { restaurant_id, name, grades }
)
```

11. Write a MongoDB query to find the restaurant name, borough, longitude and attitude and cuisine for those restaurants which contains 'mon' as three letters somewhere in its name.

```
db.restaurants.find(
  { name: { $regex: /mon/i } },
  { name, borough, "address.coord": 1, cuisine }
)
```

12. Write a MongoDB query to find the restaurant name, borough, longitude and latitude and cuisine for those restaurants which contain 'Mad' as first three letters of its name.

```
db.restaurants.find(
  { name: { $regex: /^Mad/i } },
  { name, borough, "address.coord": 1, cuisine }
)
```

13. Write a MongoDB query to find the restaurants that have at least one grade with a score of less than 5.

```
db.restaurants.find(
  {
    grades: {
      $elemMatch: { score: { $lt: 5 } }
    }
  }
)
```

14. Write a MongoDB query to find the restaurants that have at least one grade with a score of less than 5 and that are located in the borough of Manhattan.

```
db.restaurants.find(
  {
    $and: [
      { borough: "Manhattan" },
      { grades: { $elemMatch: { score: { $lt: 5 } } } }
    ]
  }
)
```

15.Write a MongoDB query to find the restaurants that have at least one grade with a score of less than 5 and that are located in the borough of Manhattan or Brooklyn.

```
db.restaurants.find(
  {
   $and: [
     { borough: { $in: ["Manhattan", "Brooklyn"] } },
     { grades: { $elemMatch: { score: { $lt: 5 } } } }
   ]
  }
)
```

16. Write a MongoDB query to find the restaurants that have at least one grade with a score of less than 5 and that are located in the borough of Manhattan or Brooklyn, and their cuisine is not American.

```
db.restaurants.find(
  {
   $and: [
     { borough: { $in: ["Manhattan", "Brooklyn"] } },
     { cuisine: { $ne: "American" } },
     { grades: { $elemMatch: { score: { $lt: 5 } } } }
   ]
  }
)
```

17. Write a MongoDB query to find the restaurants that have at least one grade with a score of less than 5 and that are located in the borough of Manhattan or Brooklyn, and their cuisine is not American or Chinese.

```
db.restaurants.find(
  {
   $and: [
     { borough: { $in: ["Manhattan", "Brooklyn"] } },
     { cuisine: { $nin: ["American", "Chinees"] } },
     { grades: { $elemMatch: { score: { $lt: 5 } } } }
   ]
  }
)
```

18. Write a MongoDB query to find the restaurants that have a grade with a score of 2 and a grade with a score of 6.

```
db.restaurants.find(
  {
   $and: [
     { grades: { $elemMatch: { score: 2 } } },
     { grades: { $elemMatch: { score: 6 } } }
   ]
  }
)
```

19. Write a MongoDB query to find the restaurants that have a grade with a score of 2 and a grade with a score of 6 and are located in the borough of Manhattan.

```
db.restaurants.find(
  {
    $and: [
      { borough: "Manhattan" },
      { grades: { $elemMatch: { score: 2 } } },
      { grades: { $elemMatch: { score: 6 } } }
    ]
  }
)
```

20. Write a MongoDB query to find the restaurants that have a grade with a score of 2 and a grade with a score of 6 and are located in the borough of Manhattan or Brooklyn.

```
db.restaurants.find(
  {
    $and: [
      { borough: { $in: ["Manhattan", "Brooklyn"] } },
      { grades: { $elemMatch: { score: 2 } } },
      { grades: { $elemMatch: { score: 6 } } }
    ]
  }
)
```

21. Write a MongoDB query to find the restaurants that have a grade with a score of 2 and a grade with a score of 6 and are located in the borough of Manhattan or Brooklyn, and their cuisine is not American.

```
db.restaurants.find(
  {
    $and: [
      { borough: { $in: ["Manhattan", "Brooklyn"] } },
      { cuisine: { $ne: "American" } },
      { grades: { $elemMatch: { score: 2 } } },
      { grades: { $elemMatch: { score: 6 } } }
    ]
  }
)
```

22. Write a MongoDB query to find the restaurants that have a grade with a score of 2 and a grade with a score of 6 and are located in the borough of Manhattan or Brooklyn, and their cuisine is not American or Chinese.

```
db.restaurants.find(
  {
    $and: [
      { borough: { $in: ["Manhattan", "Brooklyn"] } },
      { cuisine: { $nin: ["American", "Chinees"] } },
      { grades: { $elemMatch: { score: 2 } } },
      { grades: { $elemMatch: { score: 6 } } }
    ]
  }
)
```

23. Write a MongoDB query to find the restaurants that have a grade with a score of 2 or a grade with a score of 6.

```
db.restaurants.find(
  {
    $or: [
      { grades: { $elemMatch: { score: 2 } } },
      { grades: { $elemMatch: { score: 6 } } }
    ]
  }
)
```

EXERCISE 20

{
_id: ObjectId("573a1390f29313caabcd42e8"),
plot: 'A group of bandits stage a brazen train hold-up, only to find a determined
posse hot on their heels.',
genres: ['Short', 'Western'],
runtime: 11,
cast: [
'A.C. Abadie',
"Gilbert M. 'Broncho Billy' Anderson",
'George Barnes',
'Justus D. Barnes'
],

poster: 'https://m.media-
amazon.com/images/M/MV5BMTU3NjE5NzYtYTYyNS00MDVmLWIwYjgtMmYwYWIxZ

DYyNzU2XkEyXkFqcGdeQXVyNzQzNzQxNzI@._V1_SY1000_SX677_AL_.jpg',
title: 'The Great Train Robbery',
fullplot: "Among the earliest existing films in American cinema - notable as the first
film that presented a narrative story to tell - it depicts a group of cowboy outlaws
who hold up a train and rob the passengers. They are then pursued by a Sheriff's
posse. Several scenes have color included - all hand tinted.",
languages: ['English'],
released: ISODate("1903-12-01T00:00:00.000Z"),
directors: ['Edwin S. Porter'],
rated: 'TV-G',
awards: { wins: 1, nominations: 0, text: '1 win.' },
lastupdated: '2015-08-13 00:27:59.177000000',
year: 1903,
imdb: { rating: 7.4, votes: 9847, id: 439 },

countries: ['USA'],
type: 'movie',
tomatoes: {
viewer: { rating: 3.7, numReviews: 2559, meter: 75 },
fresh: 6,
critic: { rating: 7.6, numReviews: 6, meter: 100 },
rotten: 0,
lastUpdated: ISODate("2015-08-08T19:16:10.000Z")
}

1.Find all movies with full information from the 'movies' collection that released in the year 1893.

```
db.movies.find(
  { year: 1893 }
)
```

2. Find all movies with full information from the 'movies' collection that have a runtime greater than 120 minutes.

```
db.movies.find(
  { runtime: { $gt: 120 } }
)
```

3. Find all movies with full information from the 'movies' collection that have "Short" Genre.

```
db.movies.find(
  { genres: "Short" }
)
```

4. Retrieve all movies from the 'movies' collection that were directed by "William K.L. Dickson" and include complete information for each movie.

```
db.movies.find(
  { directors: "William K.L. Dickson" }
)
```

5. Retrieve all movies from the 'movies' collection that were released in the USA and include complete information for each movie.

```
db.movies.find(
  { countries: "USA" }
)
```

6. Retrieve all movies from the 'movies' collection that have complete information and are rated as "UNRATED".

```
db.movies.find(
  { rated: "UNRATED" }
)
```

7. Retrieve all movies from the 'movies' collection that have complete information and have received more than 1000 votes on IMDb.

```
db.movies.find(
  { "imdb.votes": { $gt: 1000 } }
)
```

8. Retrieve all movies from the 'movies' collection that have complete information and have an IMDb rating higher than 7.

```
db.movies.find(
  { "imdb.rating": { $gt: 7 } }
)
```

9. Retrieve all movies from the 'movies' collection that have complete information and have a viewer rating higher than 4 on Tomatoes.

```
db.movies.find(
  { "tomatoes.viewer.rating": { $gt: 4 } }
)
```

10. Retrieve all movies from the 'movies' collection that have received an award.

```
db.movies.find(
  { "awards.wins": { $gt: 0 } }
)
```

11. Find all movies with title, languages, released, directors, writers, awards, year, genres, runtime, cast, countries from the 'movies' collection in MongoDB that have at least one nomination.

```
db.movies.find(
  { "awards.nominations": { $gt: 0 } },
  { title, languages, released, directors, writers, awards, year, genres, runtime, cast, countries }
)
```

12. Find all movies with title, languages, released, directors, writers, awards, year, genres, runtime, cast, countries from the 'movies' collection in MongoDB with cast including "Charles Kayser".

```
db.movies.find(
  { cast: "Charles Kayser" },
  { title, languages, released, directors, writers, awards, year, genres, runtime, cast, countries }
)
```

13. Retrieve all movies with title, languages, released, directors, writers, countries from the 'movies' collection in MongoDB that released on May 9, 1893.

```
db.movies.find(
  { released: ISODate("1893-05-09T00:00:00Z") },
  { title, languages, released, directors, writers, countries }
)
```

14. Retrieve all movies with title, languages, released, directors, writers, countries from the 'movies' collection in MongoDB that have a word "scene" in the title.

```
db.movies.find(
  { title: { $regex: /scene/i } },
  { title, languages, released, directors, writers, countries }
)
```