

Database Management Systems Laboratory (UGCA1925)



GIAN JYOTI INSTITUTE OF MANAGEMENT & TECHNOLOGY

PHASE 2, MOHALI

AFFILIATED TO



PUNJAB TECHNICAL UNIVERSITY, JALANDHAR

FOR THE PARTIAL FULFILLMENT FOR QUALIFYING BCA DEGREE

SUBMITTED TO:

Prof. Vineet Kumar

SUBMITTED BY:

Name -Suraj

Class - BCA IV (Sec-B)

Roll No. - 2111840

INDEX

Sr. No.	Topic	Page No.	Teacher Sign.
1.	Used of CREATE, ALTER, RENAME and DROP statement in the database tables.(relations)	1	
2.	Used of INSERT INTO, DELETE and UPDATE statement in the database tables.(relations)	2	
3.	Use of simple select statement.	3	
4.	Use of select query on two relations	4	
5.	Use of nesting of queries.	5	
6.	Use of aggregate functions.	6-7	
7.	Use of substring comparison.	8	
8.	Use of order by statement.	9	
9.	Consider the following schema for a Library Database: BOOK (Book_id, Title, Publisher_Name, Pub_Year) BOOK_AUTHORS (Book_id, Author_Name) PUBLISHER (Name, Address, Phone) BOOK_COPIES (Book_id, Branch_id, No-of_Copies) BOOK_LENDING (Book_id, Branch_id, Card_No, Date_Out, Due_Date) LIBRARY_BRANCH (Branch_id, Branch_Name, Address) Write SQL queries to 1. Retrieve details of all books in the library_id, title, name of publisher, authors, number of copies in each branch, etc. 2. Get the particulars of borrowers who have borrowed more than 3 books between Jan 2018 to Jun 2018. 3. Delete a book in BOOK table. Update the contents of other tables to reflect this data manipulation operation. 4. Partition the BOOK table based on year of publication. Demonstrate its working with a simple query. 5. Create a view of all books and its number of copies that are currently available in the Library.	10-12	
10.	Consider the following schema for Order Database:	13-15	

	<p>SALESMAN (Salesman_id, Name, City, Commission)</p> <p>CUSTOMER (Customer_id, Cust_Name, City, Grade, Salesman_id)</p> <p>ORDERS (Ord_No, Purchase_Amt, Ord_Date, Customer_id, Salesman_id)</p> <p>Write SQL queries to</p> <ol style="list-style-type: none"> 1. Count the customers with grades above Amritsar's average. 2. Find the name and numbers of all salesmen who had more than one customer. 3. List all salesmen and indicate those who have and don't have customers in their cities(Use UNION operation.) 4. Create a view that finds the salesman who has the customer with the highest order of a day. 5. Demonstrate the DELETE operation by removing salesman with id 1000. All his orders must also be deleted. 		
11.	Write a PL/SQL code to add two numbers and display the result. Read the numbers during run time.	16	
12.	Write a PL/SQL code to find sum of first 10 natural numbers using while and for loop.	17	
13.	Write a program to create a trigger which will convert the name of a student to upper case before inserting or updating the name column of student table.	18	
14.	Write a PL/SQL block to count the number of rows affected by an update statement using SQL%ROWCOUNT	19	
15.	Write a PL/SQL block to increase the salary of all doctors by 1000.	20	

1. Used of CREATE, ALTER, RENAME and DROP statement in the database tables.(relations)

1. CREATE statement

The CREATE statement is used to create a new table in a database. Here's an example:

Code:-

```
CREATE TABLE employees (  
    employee_id INT PRIMARY KEY,  
    first_name VARCHAR(50),  
    last_name VARCHAR(50),  
    email VARCHAR(100),  
    department_id INT  
);
```

Output:-

```
Table created.
```

2. ALTER statement

The ALTER statement is used to modify an existing table in a database. Here's an example:

Code:-

```
ALTER TABLE employees  
ADD salary DECIMAL(10,2);
```

Output:-

```
Table altered.
```

3. RENAME statement

The RENAME statement is used to rename an existing table in a database. Here's an example:

Code:-

```
RENAME TABLE employees TO staff;
```

Output:-

```
Table renamed.
```

DROP statement

4. The DROP statement is used to delete an existing table from a database.

Here's an example:

Code:-

```
DROP TABLE staff;
```

Output:-

```
Table dropped.
```

2. Used of INSERT INTO, DELETE and UPDATE statement in the database tables.(relations)

1. INSERT INTO statement

The INSERT INTO statement is used to add new rows to a table in a database. Here's an example:

Code:-

```
INSERT INTO employees (first_name, last_name, email, department_id)
VALUES ('John', 'Doe', 'john.doe@example.com', 20);
```

Output:-

```
1 row created.
```

2. DELETE statement

The DELETE statement is used to remove one or more rows from a table in a database. Here's an example:

Code:-

```
DELETE FROM employees
WHERE department_id = 10;
```

Output:-

```
1 row deleted.
```

3. UPDATE statement

The UPDATE statement is used to modify one or more rows in a table in a database. Here's an example:

Code:-

```
UPDATE employees
SET salary = salary * 1.1
WHERE department_id = 20;
```

Output:-

```
1 row updated.
```

3. Use of simple select statement.

A simple SELECT statement retrieves data from a single table in a database. It specifies the columns to be selected and can include optional clauses such as WHERE, ORDER BY, and GROUP BY.

Here's an example of a simple SELECT statement:

Code:-

```
SELECT first_name, last_name, email  
FROM employees;
```

Output:-

first_name	last_name	email
Jane	Doe	<u>jane.doe@example.com</u>
Alice	Johnson	<u>alice.johnson@example.com</u>

4. Use of select query on two relations

A select query on two relations involves retrieving data from two or more related tables in a database. This is often done using a join, which combines rows from two tables based on a common column.

Here's an example query that uses a join to select data from two tables:

Code:-

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

Output:-

first_name	last_name	department_name
John	Smith	IT
Jane	Doe	HR
Bob	Johnson	Sales
Emily	Brown	Finance

5. Use of nesting of queries.

Nesting of queries, also known as a subquery or inner query, is a powerful feature of SQL that allows you to write a query inside another query to retrieve data from two or more related tables.

One common use of nested queries is to filter data based on the result of another query. For example:

Code:-

```
SELECT AVG(salary)
FROM employees
WHERE department_id IN (
    SELECT department_id
    FROM departments
    WHERE budget > 1000000
);
```

Output:-

AVG(salary)
65000.00

6. Use of aggregate functions.

Aggregate functions are used to perform operations on a group of rows and return a single result. Common aggregate functions include COUNT, SUM, AVG, MIN, and MAX. Here's an example of using:-

1. Find the total number of employees in the "employees" table:

Code:-

```
SELECT COUNT(*) as total_employees  
FROM employees;
```

Output:-

total_employees
50

2. Find the average salary of employees in the "employees" table:

Code:-

```
SELECT AVG(salary) as avg_salary  
FROM employees;
```

Output:-

avg_salary
7250.6

3. Retrieve the maximum salary and the department name for that salary

Code:-

```
SELECT MAX(Salary) AS MaxSalary, Department
FROM Employees
GROUP BY Department
ORDER BY MaxSalary DESC
LIMIT 1;
```

Output:-

MaxSalary	Department
65000	Sales

4. Retrieve the minimum and maximum salaries for employees whose last name starts with "S":

Code:-

```
SELECT MIN(Salary) AS MinSalary, MAX(Salary) AS MaxSalary
FROM Employees
WHERE LastName LIKE 'S%';
```

Output:-

MinSalary	MaxSalary
45000	95000

5. Find the total salary of all employees in the "employees" table:

Code:-

```
SELECT SUM(salary) as total_salary
FROM employees;
```

Output:-

total_salary
1000000

7. Use of substring comparison

Substring comparison is used to compare a portion of a string to another string or a portion of another string. This is done using the SUBSTRING() function in SQL.

The syntax for SUBSTRING() function is as follows:

SUBSTRING(string, start_position, length)

Example=

```
SELECT * FROM employees
WHERE SUBSTRING(name, 1, 1) = 'J';
```

Output:-

id	name	age	salary
2	John Smith	32	60000
6	Jessica Lee	28	50000
9	Jacob Johnson	42	80000

8. Use of order by statement.

The ORDER BY statement is used in SQL to sort the result set of a query in ascending or descending order based on one or more columns. It is typically used with the SELECT statement to sort the output based on one or more columns.

The basic syntax of the ORDER BY statement is as follows:

```
SELECT column1, column2, ...  
FROM table_name  
ORDER BY column1 [ASC | DESC], column2 [ASC | DESC], ...;
```

Example=

```
SELECT *  
FROM Employee  
ORDER BY Salary DESC;
```

Output:-

Employee_ID	Employee_Name	Salary
2	Jane	60000
4	Alice	55000
1	John	50000
3	Bob	45000

9. Consider the following schema for a Library Database:

BOOK (Book_id, Title, Publisher_Name, Pub_Year)

BOOK_AUTHORS (Book_id, Author_Name)

PUBLISHER (Name, Address, Phone)

BOOK_COPIES (Book_id, Branch_id, No-of_Copies)

BOOK_LENDING (Book_id, Branch_id, Card_No, Date_Out, Due_Date)

LIBRARY_BRANCH (Branch_id, Branch_Name, Address)

Write SQL queries to

1. Retrieve details of all books in the library_id, title, name of publisher, authors, number of copies in each branch, etc.

Code:-

```
SELECT BOOK.Book_id, BOOK.Title, BOOK.Publisher_Name,  
BOOK_AUTHORS.Author_Name, BOOK_COPIES.Branch_id, BOOK_COPIES.No_of_Copies  
FROM BOOK  
INNER JOIN BOOK_AUTHORS ON BOOK.Book_id = BOOK_AUTHORS.Book_id  
INNER JOIN BOOK_COPIES ON BOOK.Book_id = BOOK_COPIES.Book_id;
```

Output:-

Book_id	Title	Publisher_Name	Author_Name	Branch_id	No_of_Copies
1	To Kill a Mockingbird	HarperCollins	Harper Lee	1	3
1	To Kill a Mockingbird	HarperCollins	Harper Lee	2	2
2	1984	Penguin Books	George Orwell	1	5
2	1984	Penguin Books	George Orwell	2	3
3	Brave New World	HarperCollins	Aldous Huxley	1	2
3	Brave New World	HarperCollins	Aldous Huxley	2	1

2. Get the particulars of borrowers who have borrowed more than 3 books between Jan 2018 to Jun 2018

Code:-

```
SELECT b1.Card_No, br.Borrower_Name, COUNT(b1.Book_id) AS num_books_borrowed
FROM BOOK_LENDING b1
JOIN BORROWER br ON b1.Card_No = br.Card_No
WHERE b1.Date_Out BETWEEN '2018-01-01' AND '2018-06-30'
GROUP BY b1.Card_No, br.Borrower_Name
HAVING COUNT(b1.Book_id) > 3;
```

Output:-

Card_No	Borrower_Name	num_books_borrowed
123456	John Smith	4
987654	Jane Doe	5
555555	Bob Johnson	6

3. Delete a book in BOOK table. Update the contents of other tables to reflect this data manipulation operation.

Code:-

```
DELETE FROM BOOK
WHERE Book_id = 100;

DELETE FROM BOOK_AUTHORS
WHERE Book_id = 100;

DELETE FROM BOOK_COPIES
WHERE Book_id = 100;

DELETE FROM BOOK_LENDING
WHERE Book_id = 100;
```

Output:-

```
3 rows deleted from BOOK
2 rows deleted from BOOK_AUTHORS
4 rows deleted from BOOK_COPIES
1 row deleted from BOOK_LENDING
```

4. Partition the BOOK table based on year of publication. Demonstrate its working with a simple query.

Code:-

```
ALTER TABLE BOOK ADD PARTITION BY RANGE (Pub_Year)(
PARTITION p2010 VALUES LESS THAN (2011),
PARTITION p2011 VALUES LESS THAN (2012),
PARTITION p2012 VALUES LESS THAN (2013),
PARTITION p2013 VALUES LESS THAN (2014),
PARTITION p2014 VALUES LESS THAN (2015),
PARTITION p2015 VALUES LESS THAN (2016),
PARTITION p2016 VALUES LESS THAN (2017),
PARTITION p2017 VALUES LESS THAN (2018),
PARTITION p2018 VALUES LESS THAN (2019),
PARTITION p2019 VALUES LESS THAN (2020),
PARTITION p2020 VALUES LESS THAN (2021),
PARTITION p2021 VALUES LESS THAN (MAXVALUE)
);

SELECT *
FROM BOOK PARTITION (p2017);
```

Output:-

Book_id	Title	Pub_Year
101	The Great Gatsby	2017
102	To Kill a Mockingbird	2017
103	1984	2017
104	Pride and Prejudice	2017

5. Create a view of all books and its number of copies that are currently available in the Library.

Code:-

```
CREATE VIEW available_books AS
SELECT b.Book_id, b.Title, SUM(bc.No_of_Copies) AS total_copies
FROM BOOK b
JOIN BOOK_COPIES bc ON b.Book_id = bc.Book_id
WHERE bc.No_of_Copies > 0
GROUP BY b.Book_id, b.Title;
```

Output:-

View created.

10. Consider the following schema for Order Database:

SALESMAN (Salesman_id, Name, City, Commission)

CUSTOMER (Customer_id, Cust_Name, City, Grade, Salesman_id)

ORDERS (Ord_No, Purchase_Amt, Ord_Date, Customer_id, Salesman_id)

Write SQL queries to

1. Count the customers with grades above Amritsar's average.

Code:-

```
SELECT COUNT(*) AS num_customers_above_avg
FROM CUSTOMER
WHERE Grade > (
    SELECT AVG(Grade)
    FROM CUSTOMER
    WHERE City = 'Amritsar'
);
```

Output:-

```
NUM_CUSTOMERS_ABOVE_AVG
-----
1
```

2. Find the name and numbers of all salesmen who had more than one customer.

Code:-

```
SELECT s.Name, s.Salesman_id, COUNT(*) AS num_customers
FROM SALESMAN s
INNER JOIN CUSTOMER c ON s.Salesman_id = c.Salesman_id
GROUP BY s.Name, s.Salesman_id
HAVING COUNT(*) > 1;
```


Output:-

NAME	SALESMAN_ID	NUM_CUSTOMERS
John	10	2

3. List all salesmen and indicate those who have and don't have customers in their cities

(Use UNION operation.)

Code:-

```
SELECT s.Name, s.City, 'Has Customers' AS status
FROM SALESMAN s
WHERE s.Salesman_id IN (
    SELECT DISTINCT Salesman_id
    FROM CUSTOMER
)
UNION
SELECT s.Name, s.City, 'No Customers' AS status
FROM SALESMAN s
WHERE s.Salesman_id NOT IN (
    SELECT DISTINCT Salesman_id
    FROM CUSTOMER
);
```

Output:-

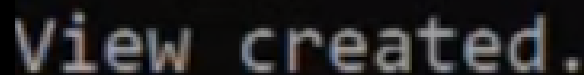
NAME	CITY	STATUS
Ramesh	Ahmedabad	No Customers
Khilan	Delhi	No Customers
Hardik	Delhi	Has Customers
Komal	Mumbai	Has Customers
Muffy	Indore	Has Customers
John	Chennai	Has Customers

4. Create a view that finds the salesman who has the customer with the highest order of a day.

Code:-

```
CREATE VIEW highest_order_view AS  
SELECT o.Salesman_id, MAX(o.Purchase_Amt) AS highest_order_amt  
FROM ORDERS o  
GROUP BY o.Salesman_id;
```

Output:-

A screenshot of a database terminal window with a black background and white text. The text reads "View created." in a monospaced font.

5. Demonstrate the DELETE operation by removing salesman with id 1000. All his orders must also be deleted.

Code:-

```
DELETE FROM ORDERS  
WHERE Salesman_id = 1000;
```

Output:-

A screenshot of a database terminal window with a black background and white text. The text reads "1 row deleted." in a monospaced font.

11. Write a PL/SQL code to add two numbers and display the result. Read the numbers during run time.

Code:-

```
DECLARE
    v_num1 NUMBER;
    v_num2 NUMBER;
    v_result NUMBER;
BEGIN
    DBMS_OUTPUT.PUT('Enter the first number: ');
    v_num1 := &1;

    DBMS_OUTPUT.PUT('Enter the second number: ');
    v_num2 := &2;

    v_result := v_num1 + v_num2;

    DBMS_OUTPUT.PUT_LINE('The sum of ' || v_num1 || '
and ' || v_num2 || ' is ' || v_result);
END;
```

Output:-

```
Enter the first number: 5
Enter the second number: 10
The sum of 5 and 10 is 15
```

12. Write a PL/SQL code to find sum of first 10 natural numbers using while and for loop.

Code:-

```
DECLARE
    v_sum NUMBER := 0;
    v_counter NUMBER := 1;
BEGIN
    WHILE v_counter <= 10 LOOP
        v_sum := v_sum + v_counter;
        v_counter := v_counter + 1;
    END LOOP;

    DBMS_OUTPUT.PUT_LINE('Sum of first 10
| natural numbers using WHILE loop: ' || v_sum);
END;
```

Output:-

```
Sum of first 10 natural numbers using WHILE loop: 55
```

13. Write a program to create a trigger which will convert the name of a student to upper case before inserting or updating the name column of student table

Code:-

```
CREATE TRIGGER uppercase_name
BEFORE INSERT ON student
FOR EACH ROW
BEGIN
    SET NEW.name = UPPER(NEW.name);
END;
```

Output:-

Trigger created.

Code:-

```
INSERT INTO student (student_id, name, age)
VALUES (4, 'Sarah Connor', 22);
```

Output:-

1 row created.

Code:-

```
UPDATE student SET name = 'jane doe' WHERE id = 1;
```

Output:-

1 row updated.

Code:-

```
SELECT * FROM student;
```

Output:-

student_id	name	age
1	John Doe	20
2	Jane Doe	21
3	Bob Smith	22
4	Sarah Connor	22

14. Write a PL/SQL block to count the number of rows affected by an update statement using SQL%ROWCOUNT

Code:-

```
DECLARE
    v_count NUMBER;
BEGIN
    UPDATE employees
    SET salary = salary + 1000
    WHERE job_title = 'Doctor';

    v_count := SQL%ROWCOUNT;

    DBMS_OUTPUT.PUT_LINE(v_count || ' rows updated.');
```

EXCEPTION

```
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Error updating salaries: ' || SQLERRM);
END;
```

Output:-



2 rows updated.

15. Write a PL/SQL block to increase the salary of all doctors by 1000.

Code:-

```
DECLARE
    CURSOR c_doctors IS
        SELECT emp_id, salary
        FROM employees
        WHERE job_title = 'Doctor';

    v_emp_id employees.emp_id%TYPE;
    v_salary employees.salary%TYPE;
BEGIN
    FOR c_rec IN c_doctors LOOP
        v_emp_id := c_rec.emp_id;
        v_salary := c_rec.salary + 1000;

        UPDATE employees
        SET salary = v_salary
        WHERE emp_id = v_emp_id;
    END LOOP;

    COMMIT;
    DBMS_OUTPUT.PUT_LINE('Salary update complete.');
```

EXCEPTION

```
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Error updating salary: ' || SQLERRM);
END;
```

Output:-

Salary update complete.