……Final……

……Lab-4……

Displaying Data from Multiple Tables----------

Cartesian Product:

```
SELECT last_name, department_name dept_name
FROM emps, depts;
```

Equijoin:

```
SELECT e.employee_id, e.last_name, e.department_id,
    d.department_id, d.location_id
FROM emps e, depts d
WHERE e.department_id = d.department_id;
```

Self-Join (Joining a table to itself):

```
SELECT worker.last_name, manager.last_name
FROM emps worker, emps manager
WHERE worker.manager_id = manager.employee_id;
```

Join using ON clause:

```
SELECT e.employee_id, e.last_name, e.department_id,
    d.department_id, d.location_id
FROM emps e
JOIN depts d ON (e.department_id = d.department_id);
```

Three-Way Join:

```
SELECT e.employee_id,l.city, d.department_name
FROM emps e
JOIN depts d ON(e.department_id= d.department_id)
JOIN locs l on (d.location_id= l.location_id);
```

LEFT OUTER JOIN:

```
SELECT e.last_name, e.department_id, d.department_name
 FROM emps e LEFT OUTER JOIN depts d
ON (e.department_id = d.department_id);
```

RIGHT OUTER JOIN:

```
SELECT e.last_name, e.department_id, d.department_name
FROM emps e RIGHT OUTER JOIN depts d
ON (e.department_id = d.department_id);
```

FULL OUTER JOIN:

```
SELECT e.last_name, e.department_id, d.department_name
FROM emps e FULL OUTER JOIN depts d
ON (e.department_id = d.department_id);
```

Join with Additional Condition:

```
SELECT e.employee_id, e.last_name, e.department_id, d.department_id, d.location_id
FROM emps e
JOIN depts d ON (e.department_id = d.department_id)
AND e.manager_id = 149;
```

Group Functions----------

AVG, MAX, MIN, SUM:

```
SELECT AVG(salary), MAX(salary), MIN(salary), SUM(salary)
FROM emps
WHERE job_id LIKE '%REP%';
```

COUNT:

```
SELECT COUNT(DISTINCT department_id)
FROM emps;
```

GROUP BY Clause----------

Single Column:

```
SELECT department_id, AVG(salary)
FROM emps
GROUP BY department_id;
```

Multiple Columns:

```
SELECT department_id, dept_id, job_id, SUM(salary)
FROM emps
GROUP BY department_id, job_id;
```

HAVING Clause:

```
SELECT job_id, SUM(salary) PAYROLL
FROM emps
WHERE job_id NOT LIKE '%REP%'
GROUP BY job_id
HAVING SUM(salary) > 13000
ORDER BY SUM(salary);
```

Nested Group Functions:

```
SELECT MAX(AVG(salary))
FROM emps
GROUP BY department_id
```

……..Lab-5 ……

UsingaSubquery

```
SELECT last_name
FROM emps
WHERE salary >(SELECT salary
FROM emps
WHERE last_name = ’Abel’);
```

Executing Single-Row Subqueries

```
SELECT last_name, job_id, salary
FROM emps
WHERE job_id =
(SELECT job_id
FROM emps
WHERE employee_id = 141)
AND salary >
(SELECT salary
FROM emps
WHERE employee_id = 143);
```

Using Group Functions in a Subquery

```
SELECT last_name, job_id, salary
FROM emps
WHERE salary =
(SELECT MIN(salary)
FROM emps);
```

Operator	Meaning
IN	Equal to any member in the list
ANY	Compare value to each value returned by the subquery
ALL	Compare value to every value returned by the subquery

Using the ANY Operator

```
SELECT employee_id, last_name, job_id, salary
FROM emps
WHERE salary < ANY
(SELECT salary
FROM emps
WHERE job_id = 'IT_PROG')
AND job_id <> 'IT_PROG';
```

Using the ALL Operator

```
SELECT employee_id, last_name, job_id, salary
FROM emps
WHERE salary < ALL
(SELECT salary
FROM emps
WHERE job_id = 'IT_PROG')
AND job_id <> 'IT_PROG';
```

Copying Rows from Another Table

```
INSERT INTO sales_reps(id, name, salary, commission_pct)
SELECT employee_id, last_name, salary, commission_pct
```

```
FROM emps
WHERE job_id LIKE '%REP%';
```

Updating Rows in a Table

```
UPDATE emps
SET department_id = 70
WHERE employee_id = 113;
```

Updating Rows Based on Another Table

```
UPDATE copy_emp
SET department_id =(SELECT department_id
FROM emps
WHERE employee_id = 100)
WHERE job_id
= (SELECT job_id
FROM emps
WHERE employee_id = 200);
```

Example of Merging Rows

```
MERGE INTO copy_emp c
USING emps e
ON (c.employee_id = e.employee_id)
WHEN MATCHED THEN
UPDATE SET
c.first_name = e.first_name,
c.last_name = e.last_name,
c.email = e.email,
c.phone_number = e.phone_number,
c.hire_date = e.hire_date,
c.job_id = e.job_id,
c.salary = e.salary,
c.commission_pct = e.commission_pct,
c.manager_id = e.manager_id,
c.department_id = e.department_id
WHEN NOT MATCHED THEN
INSERT VALUES(e.employee_id, e.first_name, e.last_name,
e.email, e.phone_number, e.hire_date, e.job_id,
e.salary, e.commission_pct, e.manager_id,
e.department_id);
```

The ALTER TABLE Statement

Use the ALTER TABLE statement to:

Addanewcolumn
Modify an existing column
Define a default value for the new column
Dropacolumn

Adding a Column

```
ALTER TABLE dept80
ADD (job_id VARCHAR2(9));
```

Modifying a Column

```
ALTER TABLE dept80
MODIFY (last_name VARCHAR2(30));
```

Dropping a Column

```
ALTER TABLE dept80
DROP COLUMN job_id;
```

Dropping a Table

```
DROP TABLE dept80;
```

Changing the Name of an Object

```
RENAME dept TO detail_dept;
```

Truncating a Table

```
TRUNCATE TABLE detail_dept;
```

Add PRIMARY KEY/ FOREIGN KEY constraints

```
ALTER TABLE emps
ADD CONSTRAINT emp_manager_fk
FOREIGN KEY(manager_id)
REFERENCES emps(employee_id);
```

Creating a View

- **Create a view by using column aliases in the subquery.**

```
CREATE VIEW  salvu50
AS SELECT  employee_id ID_NUMBER, last_name NAME,
           salary*12 ANN_SALARY
   FROM    employees
   WHERE   department_id = 50;
View created.
```

……Lab 06……

```
CREATE DATABASE `Book`;
USE `Book`;


CREATE TABLE `Publisher` (
   Publisher_Name VARCHAR(100) PRIMARY KEY,
   Address VARCHAR(200),
   Phone BIGINT
);


CREATE TABLE `Book` (
   Book_id INT PRIMARY KEY,
   Title VARCHAR(200),
   Pub_Year VARCHAR(20),
   Publisher_Name VARCHAR(100),
   FOREIGN KEY (Publisher_Name) REFERENCES `Publisher`(Publisher_Name) ON
DELETE CASCADE
);


CREATE TABLE `Book_Authors` (
   Author_Name VARCHAR(100),
   Book_id INT,
   PRIMARY KEY (Book_id, Author_Name),
   FOREIGN KEY (Book_id) REFERENCES `Book`(Book_id) ON DELETE CASCADE
```

```
);


CREATE TABLE `Library_Branch` (
    Branch_id INT PRIMARY KEY,
    Branch_Name VARCHAR(100),
    Address VARCHAR(200)
);


CREATE TABLE `Book_Copies` (
    No_of_copies INT,
    Book_id INT,
    Branch_id INT,
    PRIMARY KEY (Book_id, Branch_id),
    FOREIGN KEY (Book_id) REFERENCES `Book`(Book_id) ON DELETE CASCADE,
    FOREIGN KEY (Branch_id) REFERENCES `Library_Branch`(Branch_id) ON
DELETE CASCADE
);


CREATE TABLE `Card` (
    Card_No INT PRIMARY KEY
);


CREATE TABLE `Book_Lending` (
    Date_out DATE,
    Due_date DATE,
    Book_id INT,
    Branch_id INT,
    Card_No INT,
    PRIMARY KEY (Book_id, Branch_id, Card_No),
    FOREIGN KEY (Book_id) REFERENCES `Book`(Book_id) ON DELETE CASCADE,
    FOREIGN KEY (Branch_id) REFERENCES `Library_Branch`(Branch_id) ON
DELETE CASCADE,
    FOREIGN KEY (Card_No) REFERENCES `Card`(Card_No) ON DELETE CASCADE
);
```

```
INSERT INTO `Publisher` (Publisher_Name, Phone, Address) VALUES
('MCGRAW-HILL', 9989076587, 'BANGALORE'),
('PEARSON', 9889076565, 'NEWDELHI'),
('RANDOM HOUSE', 7455679345, 'HYDERABAD'),
('HACHETTE LIVRE', 8970862340, 'CHENNAI'),
('GRUPO PLANETA', 7756120238, 'BANGALORE');


INSERT INTO `Book` (Book_id, Title, Pub_Year, Publisher_Name) VALUES
(1, 'DBMS', 'JAN-2017', 'MCGRAW-HILL'),
(2, 'ADBMS', 'JUN-2016', 'MCGRAW-HILL'),
(3, 'CN', 'SEP-2016', 'PEARSON'),
(4, 'CG', 'SEP-2015', 'GRUPO PLANETA'),
(5, 'OS', 'MAY-2016', 'PEARSON');


INSERT INTO `Book_Authors` (Author_Name, Book_id) VALUES
('NAVATHE', 1),
('NAVATHE', 2),
('TANENBAUM', 3),
('EDWARD ANGEL', 4),
('GALVIN', 5);


INSERT INTO `Library_Branch` (Branch_id, Branch_Name, Address) VALUES
(10, 'RR NAGAR', 'BANGALORE'),
(11, 'RNSIT', 'BANGALORE'),
(12, 'RAJAJI NAGAR', 'BANGALORE'),
(13, 'NITTE', 'MANGALORE'),
(14, 'MANIPAL', 'UDUPI');


INSERT INTO `Book_Copies` (No_of_copies, Book_id, Branch_id) VALUES
(10, 1, 10),
(5, 1, 11),
(2, 2, 12),
(5, 2, 13),
(7, 3, 14),
```

```
(1, 5, 10),
(3, 4, 11);


INSERT INTO `Card` (Card_No) VALUES
(100),
(101),
(102),
(103),
(104);


INSERT INTO `Book_Lending` (Date_out, Due_date, Book_id, Branch_id, Card_No)
VALUES
('2017-01-01', '2017-06-01', 1, 10, 101),
('2017-01-11', '2017-03-11', 3, 14, 101),
('2017-02-21', '2017-04-21', 2, 13, 101),
('2017-03-15', '2017-07-15', 4, 11, 101),
('2017-04-12', '2017-05-12', 1, 11, 104);


1. SELECT B.BOOK_ID, B.TITLE, B.PUBLISHER_NAME, A.AUTHOR_NAME,
C.NO_OF_COPIES, L.BRANCH_ID
FROM BOOK B, BOOK_AUTHORS A, BOOK_COPIES C, LIBRARY_BRANCH L
WHERE B.BOOK_ID = A.BOOK_ID
  AND B.BOOK_ID = C.BOOK_ID
  AND L.BRANCH_ID = C.BRANCH_ID;

2. SELECT CARD_NO
FROM BOOK_LENDING
WHERE DATE_OUT BETWEEN '2017-01-01' AND '2017-06-30'
GROUP BY CARD_NO
HAVING COUNT(*) > 3;

4. CREATE VIEW V_PUBLICATION AS
SELECT PUB_YEAR FROM BOOK;

SELECT * FROM V_PUBLICATION;
```