CHAITANYA BHARATHI INSTITUTE OF TECHNOLOGY (A)
Kokapet(Village), Gandipet, Hyderabad, Telangana-500075. www.cbit.ac.in

COMMITTED TO RESEARCH, INNOVATION AND EDUCATION 44 years

CERTIFICATE

Certified that this is the bonafide record of the practical work done during the academic year **2022-2023** by Mr. **NAMA SRICHARAN**

Bearing Roll No. **160121733117** of class **CSE-2** of semester **IV** in the Laboratory of **DATABASE MANAGEMENT SYSTEMS LAB (20CSC17)** of the department of **Computer Science and Engineering**.

Signature of the Staff Member in Charge

Signature of the External Examiner **Signature of the Internal Examiner**

INDEX

EXPERIMENT 1

AIM : To make a comparative study of different RDBMS product and NoSQL databases

DESCRIPTION :

Some of the popular RDBMS products are as follows :

- Oracle Database: Oracle is a widely used commercial RDBMS known for its robustness, scalability, and advanced features. It offers comprehensive support for transaction processing, high availability, and data integrity.
- Microsoft SQL Server: SQL Server is a commercial RDBMS developed by Microsoft. It provides a strong feature set, integration with the Microsoft ecosystem, and support for business intelligence and analytics.
- MySQL: MySQL is an open-source RDBMS known for its ease of use, performance, and scalability. It is widely used in web applications and offers a variety of storage engines, including InnoDB for transactional support.
- PostgreSQL: PostgreSQL is a powerful open-source RDBMS known for its adherence to standards, extensibility, and robustness. It offers a rich feature set, including support for complex queries, spatial data, and JSON
- IBM Db2: Db2 is a commercial RDBMS developed by IBM. It is known for its scalability, performance, and integration with other IBM products. It offers advanced features like data compression and workload management.
- SQLite: SQLite is a lightweight, file-based RDBMS that is often embedded in applications. It is known for its simplicity, zero-configuration setup, and small footprint. It is widely used in mobile and IoT applications.
- MariaDB: MariaDB is a community-developed fork of MySQL that aims to be a drop-in replacement. It offers enhanced features, improved performance, and compatibility with MySQL.
- Amazon RDS: Amazon RDS is a cloud-based service that provides managed relational databases, including options for MySQL, PostgreSQL, Oracle, and Microsoft SQL Server. It offers scalability, high availability, and automated backups.
- SAP HANA: SAP HANA is an in-memory RDBMS designed for high-performance analytics and real-time data processing. It offers advanced features like columnar storage, data compression, and predictive analytics.
- Teradata: Teradata is a commercial RDBMS known for its data warehousing capabilities and scalability. It is designed for handling large volumes of data and complex analytics workloads.

Some of the popular NOSQL databases are as follows :

- MongoDB: MongoDB is a document-oriented database that can be deployed in a distributed manner. It provides scalability and flexible data modeling with support for horizontal scaling and automatic sharding.

- Amazon DynamoDB: DynamoDB is a fully managed NoSQL database service provided by Amazon Web Services (AWS). It offers seamless scalability, high availability, and automatic replication across multiple nodes.

- Google Cloud Spanner: Spanner is a globally distributed relational database offered by Google Cloud. It provides strong consistency, horizontal scalability, and automatic data replication across multiple regions

- CockroachDB: CockroachDB is an open-source distributed SQL database designed for global scalability and high availability. It combines the scalability of NoSQL with the consistency of SQL databases.

- Apache HBase: HBase is a distributed, scalable, and column-oriented NoSQL database built on top of the Hadoop ecosystem. It provides real-time read and write access to large datasets.

- YugabyteDB: YugabyteDB is an open-source, distributed SQL database designed for global deployments. It offers compatibility with PostgreSQL and Cassandra APIs, allowing users to leverage the benefits of both databases

- ScyllaDB: ScyllaDB is a highly performant NoSQL database built as a drop-in replacement for Apache Cassandra. It leverages the scalability and fault tolerance of Cassandra while delivering improved performance

- Couchbase: Couchbase is a distributed NoSQL database that combines the capabilities of a document database with the performance of in-memory caching. It offers easy scalability, flexible data modelling, and support for high availability.

CONCLUSION : A comparative study on the various RDBMS products and NOSQL databases was successfully made.

****

EXPERIMENT 2

SQL – DDL and DML

1.DDL (DATA DEFINITION LANGUAGE) –

This language is used to create structure, alter, drop ,truncate.

a. CREATE : It is used to create table .During creation of the table we need to specify the column names and their datatypes.

Syntax: Create table employee (Eid int ,name varchar(34));

b. ALTER : It is used for adding or removing columns, renaming column, changing the datatype.

Syntax: Alter table employee
add (sal int);

c. DROP : To completely delete the structure of the table including data inside , the drop command is used.

Syntax : Drop table employee;

d. TRUNCATE : It similar to drop command except that it preserves the structure of the table

Syntax: Truncate table employee;

NOTE: The above 4 commands create, alter, drop, truncate cannot be reversed which means roll backed.

2.DML (DATA MANIPULATION LANGUAGE) :

Following are the commands of DML:

a. INSERT : It is used to insert records into the table .

Syntax : Insert into employee
values(101,'ABC'),
values(102,'xyz') ;

b. DELETE : It is used to delete a single record from the table.

Syntax : Delete from employee
Where Eid = 101;

c. UPDATE : It is used to update any new record in the table.

Syntax : Update employee
Set name = 'wxy' where Eid=102;

NOTE : Unlike DDL commands DML commands can be roll backed.

AIM : (i) Create employee table and insert few records.
(ii) Alter data type and add column in a table.
(iii) Update any column values based on where condition.
(iv) Write SQL commands to illustrate the difference between Drop, Truncate and Delete.

(i) Create emp table and insert few records :

CODE :

```
Show databases;
Use mysql;
```

Creating a table

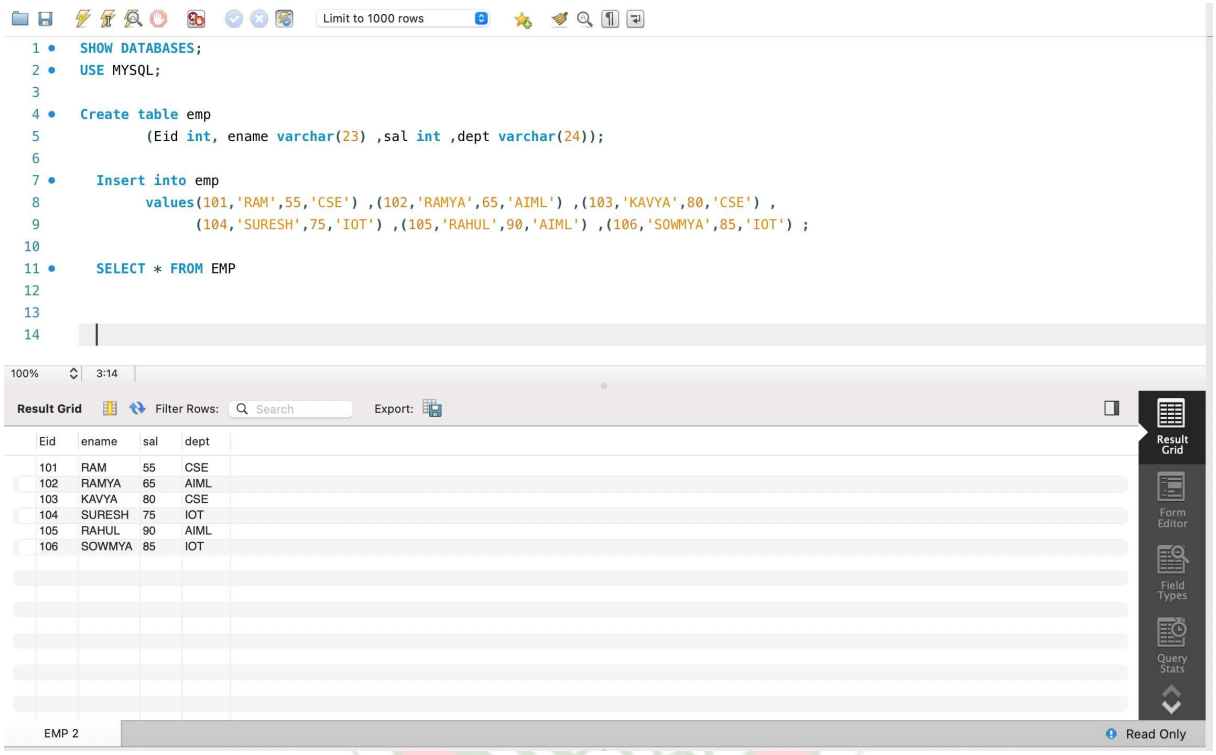
```
Create table emp
  (Eid int, ename varchar(23) ,sal int ,dept varchar(24));
```

Inserting few records into the table :

```
Insert into emp
    values(101,'A','CSE',55) ,
          (102,'B,'AIML',65) ,
          (103,'C','CSE',80) ,
          (104,'D','IOT',75) ,
          (105,'E','AIML',90) ,
          (106,'F','IOT',85);
```

DEMONSTRATION :

(ii) Alter data type and add column in a table :



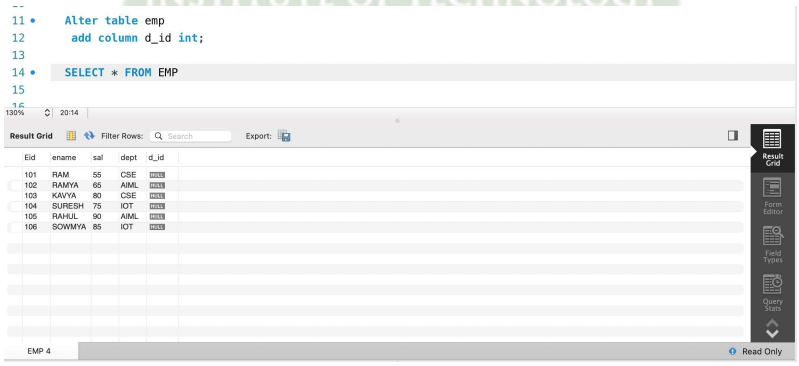
```
SHOW DATABASES;
USE MYSQL;

Create table emp
        (Eid int, ename varchar(23) ,sal int ,dept varchar(24));

  Insert into emp
        values(101,'RAM',55,'CSE') ,(102,'RAMYA',65,'AIML') ,(103,'KAVYA',80,'CSE') ,
              (104,'SURESH',75,'IOT') ,(105,'RAHUL',90,'AIML') ,(106,'SOWMYA',85,'IOT') ;

  SELECT * FROM EMP
```

Result Grid

Eid	ename	sal	dept
101	RAM	55	CSE
102	RAMYA	65	AIML
103	KAVYA	80	CSE
104	SURESH	75	IOT
105	RAHUL	90	AIML
106	SOWMYA	85	IOT

EMP 2 — Read Only

CODE: OUTPUT



```
Alter table emp
 add column d_id int;

SELECT * FROM EMP
```

Result Grid

Eid	ename	sal	dept	d_id
101	RAM	55	CSE	NULL
102	RAMYA	65	AIML	NULL
103	KAVYA	80	CSE	NULL
104	SURESH	75	IOT	NULL
105	RAHUL	90	AIML	NULL
106	SOWMYA	85	IOT	NULL

EMP 4 — Read Only

```
Alter table emp
Add column d_id int;
```

(iii) Update any column values based on where condition.

CODE:

```
Update emp
    Set d_id = 1 where Eid = 101;
```

```
Update emp
    Set d_id = 2 where Eid = 102;

Update emp
    Set d_id = 3 where Eid = 103;
```

NOTE : If you get a error message as. “You are using safe update mode and you tried to update a table without a WHERE that uses a KEY column.”

Write this command so that you can execute update command.

“SET SQL_SAFE_UPDATES = 0;”

OUTPUT:

```
17 •  Update emp
18    Set d_id = 1 where Eid = 101;
19
20 •   Update emp
21    Set d_id = 2 where Eid = 102;
22
23 •   Update emp
24    Set d_id = 3 where Eid = 103;
25
26 •   Update emp
27    Set d_id = 4 where Eid = 104;
28
29 •   Update emp
30    Set d_id = 5 where Eid = 105;
31
32 •   Update emp
33    Set d_id = 6 where Eid = 106;
34
```

100% 4:34

Result Grid Filter Rows: Search Export:

Eid	ename	sal	dept	d_id
101	RAM	55	CSE	1
102	RAMYA	65	AIML	2
103	KAVYA	80	CSE	3
104	SURESH	75	IOT	4
105	RAHUL	90	AIML	5
106	SOWMYA	85	IOT	6

EMP 7 Read Only

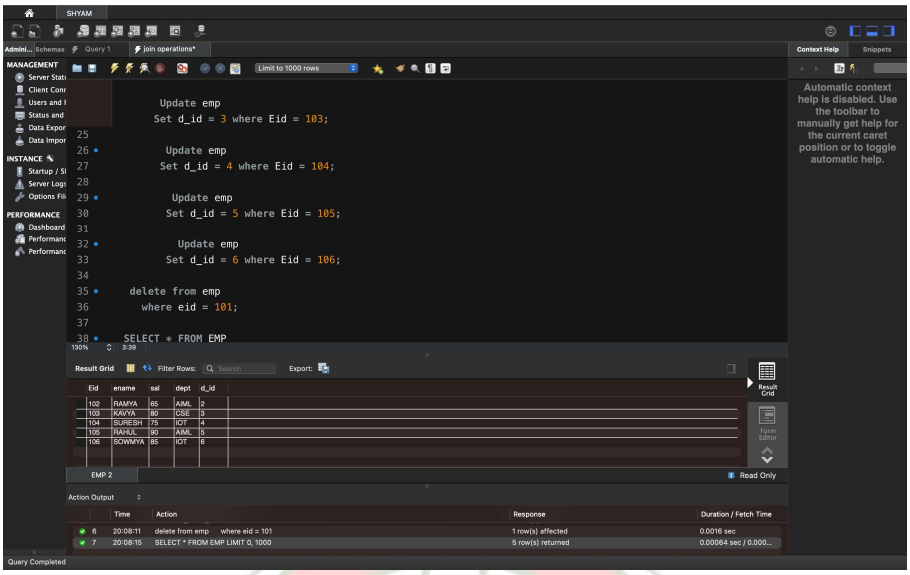
(iv) Write SQL commands to illustrate the difference between Drop, Truncate and Delete

• DELETE :

```
Delete from emp
    Where Eid = 101;
```

On executing the above command only the record you want to delete gets deleted.

OUTPUT :

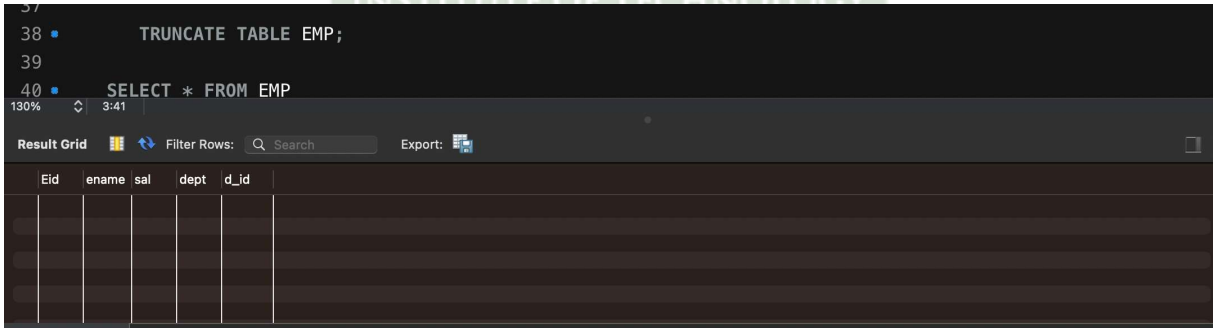


• TRUNCATE :

CODE : Truncate table emp ;

On executing the above command the entire data gets deleted except that it preserves the structure of the table.

OUTPUT :

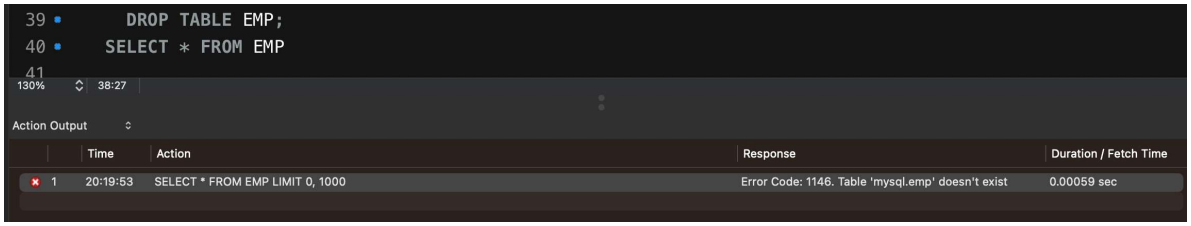


• DROP :

CODE : DROP TABLE EMP;

On executing the above command the structure of the table including the structure of the table gets dropped.

OUTPUT :



RESULT : Executed the above above queries using the various DDL and DML commands.

****

EXPERIMENT 3

WHERE CLAUSE AND SET OPERATIONS

AIM:-

Select with where clause and set operations

1. Using relational operators in where clause, find employees who are working in CSEand lives in hyderabad city

2. Using relational operators in where clause, find employees who are either working for AIML orCSE with salary > 50k

3 .SET operations

DESCRIPTION:-

- In SQL, relational operators are used to compare values and perform conditional operations in the WHERE clause of a SELECT statement. They allow you to filter and retrieve data based on specific conditions. The most commonly used relational operators in SQL are:
- Equal To (=): The equal to operator is used to check if two values are equal.
- Not Equal To (!= or <>): The not equal to operator checks if two values are not equal.
- Greater Than (>): The greater than operator checks if the left operand is greater than the right operand.
- Less Than (<): The less than operator checks if the left operand is less than the right operand.
- Greater Than or Equal To (>=): The greater than or equal to operator checks if the left operand is greater than or equal to the right operand.
- Less Than or Equal To (<=): The less than or equal to operator checks if the left operand is less than or equal to the right operand.

SET operators are special types of operators which are used to combine the result of two queries.

Operators covered under SET operators are:

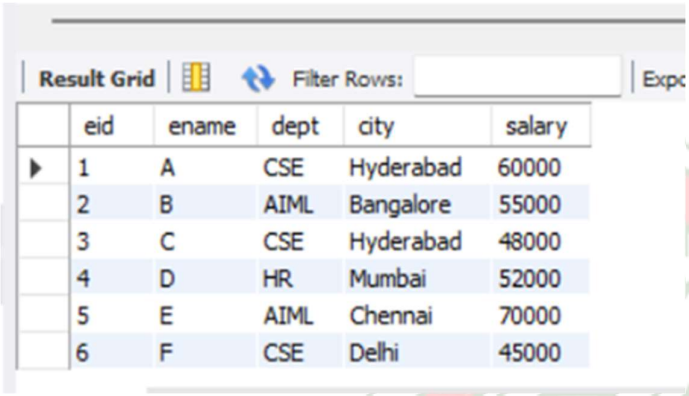
1. Union : Combines the results of two or more SELECT queries, removing duplicates.
2. Union all : The UNION ALL operation is used to combine the results of two or more SELECT queries, including duplicate rows.
3. Intersect : Returns only the common rows between two SELECT queries.
4. Minus: used to subtract the result of one SELECT query from another, returning only the rows that are unique to the first query.

CODE:

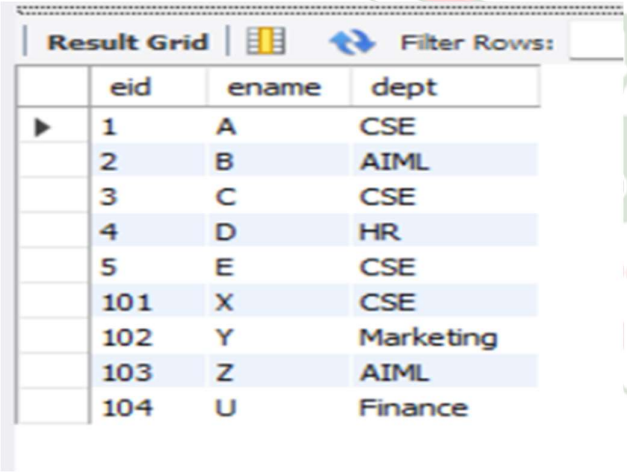
```
create table emp(
eid int,
ename varchar(45),
dept varchar(45),
city varchar(45),
salary int
);
INSERT INTO emp (eid, ename, dept, city, salary)
VALUES
   (1, 'A', 'CSE', 'Hyderabad', 60000),
   (2, 'B', 'AIML', 'Bangalore', 55000),
   (3, 'C', 'CSE', 'Hyderabad', 48000),
   (4, 'D', 'HR', 'Mumbai', 52000),
   (5, 'E', 'AIML', 'Chennai', 70000),
   (6, 'F', 'CSE', 'Delhi', 45000);
select * from emp;
SELECT * FROM emp WHERE dept = 'CSE' AND city = 'Hyderabad';
SELECT * FROM emp WHERE (dept = 'AIML' OR dept = 'CSE') AND salary > 50000;
CREATE TABLE employee (
   eid INT,
   ename VARCHAR(100),
   dept VARCHAR(50)
);
INSERT INTO employee (eid, ename, dept)
VALUES
   (1, 'A', 'CSE'),
   (2, 'B', 'AIML'),
   (3, 'C', 'CSE'),
   (4, 'D', 'HR'),
   (5, 'E', 'CSE'),
   (2, 'B', 'AIML'),
   (3, 'C', 'CSE');
CREATE TABLE contractors (
   cid INT PRIMARY KEY,
   cname VARCHAR(100),
   dept VARCHAR(50)
);
INSERT INTO contractors (cid, cname, dept)
VALUES
   (101, 'X', 'CSE'),
   (102, 'Y', 'Marketing'),
   (103, 'Z', 'AIML'),
   (104, 'U', 'Finance'),
   (1, 'A', 'CSE'),
   (2, 'B', 'AIML'),
   (3, 'C', 'CSE');
```

```
-- union
SELECT * FROM employee UNION SELECT * FROM contractors;
-- union all
SELECT * FROM employee UNION ALL SELECT * FROM contractors;
-- intersect
SELECT * FROM employee INTERSECT SELECT * FROM contractors;
-- minus(except)
SELECT * FROM employee EXCEPT SELECT * FROM contractors;
```

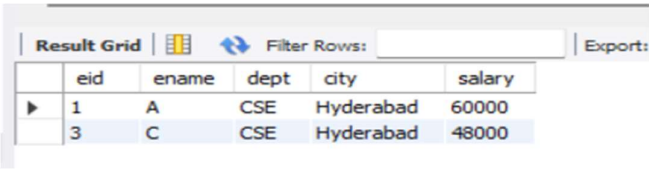
OUTPUT:-



eid	ename	dept	city	salary
1	A	CSE	Hyderabad	60000
2	B	AIML	Bangalore	55000
3	C	CSE	Hyderabad	48000
4	D	HR	Mumbai	52000
5	E	AIML	Chennai	70000
6	F	CSE	Delhi	45000



eid	ename	dept
1	A	CSE
2	B	AIML
3	C	CSE
4	D	HR
5	E	CSE
101	X	CSE
102	Y	Marketing
103	Z	AIML
104	U	Finance



eid	ename	dept	city	salary
1	A	CSE	Hyderabad	60000
3	C	CSE	Hyderabad	48000

Result Grid | Filter Rows:

	eid	ename	dept
▶	1	A	CSE
	2	B	AIML
	3	C	CSE
	4	D	HR
	5	E	CSE
	2	B	AIML
	3	C	CSE
	1	A	CSE
	2	B	AIML
	3	C	CSE
	101	X	CSE
	102	Y	Mar...
	103	Z	AIML
	104	U	Fina...

Result Grid | Filter Rows:

	eid	ename	dept
▶	1	A	CSE
	2	B	AIML
	3	C	CSE

Result Grid | Filter Rows:

	eid	ename	dept
▶	4	D	HR
	5	E	CSE

Result Grid | Filter Rows: | Exp

	eid	ename	dept	city	salary
▶	1	A	CSE	Hyderabad	60000
	2	B	AIML	Bangalore	55000
	5	E	AIML	Chennai	70000

RESULT : Executed the above above queries using the various DDL and DML commands.

EXPERIMENT 4

AGGREGATION FUNCTIONS

AIM : Using Aggregation functions find the below queries :

(i) Find max salary in AIML Dept.
(ii) Count number of employees working in AIML and CSE Depts.

DESCRIPTION :- Aggregation functions, also known as aggregate functions, are a set of functions in SQL that perform calculations on a set of rows and return a single value. These functions are often used in combination with the SELECT statement to generate summary information from a table or a group of rows. Here are some common aggregation functions:

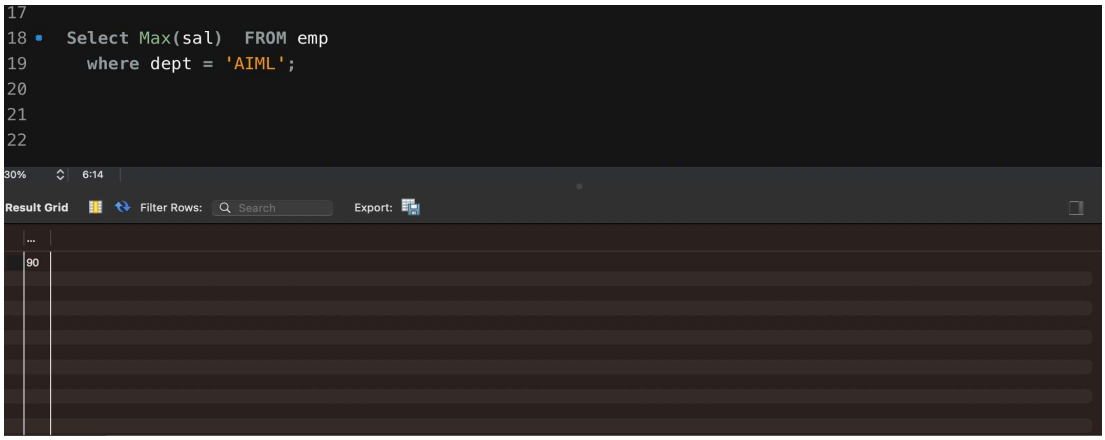
- COUNT
- SUM
- AVG
- MIN
- MAX
- GROUP BY

CODE :

(i) Find max salary in AIML Dept.

```
Select Max(sal) from emp
    Where dept = 'AIML';
```

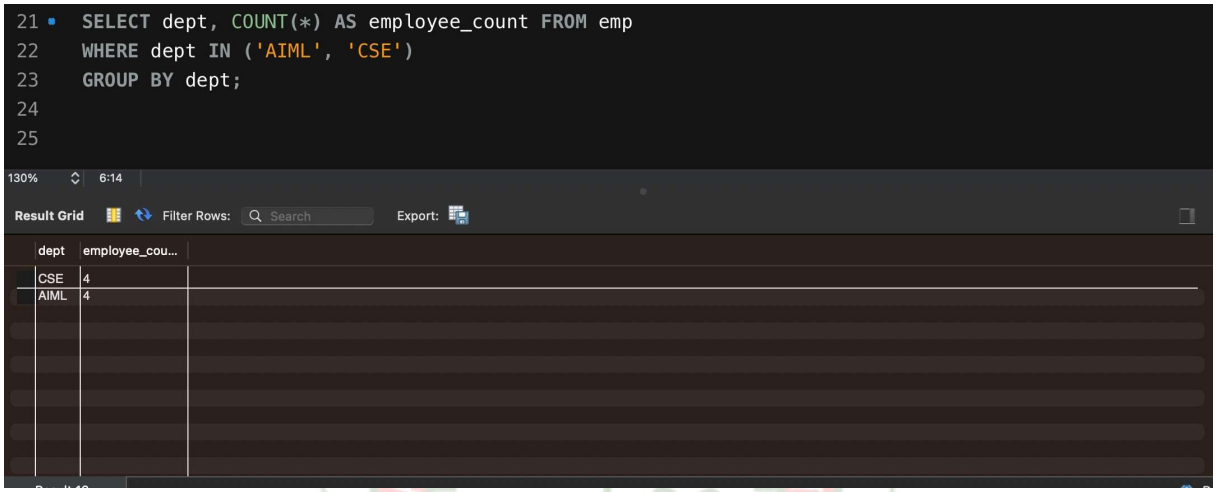
OUTPUT :



(ii). Count number of employees working in AIML and CSE Depts.

```
SELECT dept, COUNT(*) AS employee_count FROM emp
WHERE dept IN ('AIML', 'CSE')
GROUP by dept;
```

OUTPUT :



RESULT : Hence executed the above queries using Aggregate functions.

****

EXPERIMENT 5

AIM :- Using Group by and having functions address the following queries:

(i) Find dept wise avg salary.

(ii) Find number of employees in each dept.

(iii) Find depts whose average salary is less than 50k.

DESCRIPTION :- Aggregation functions, also known as aggregate functions, are a set of functions in SQL that perform calculations on a set of rows and return a single value. These functions are often used in combination with the SELECT statement to generate summary information from a table or a group of rows. Here are some common aggregation functions:

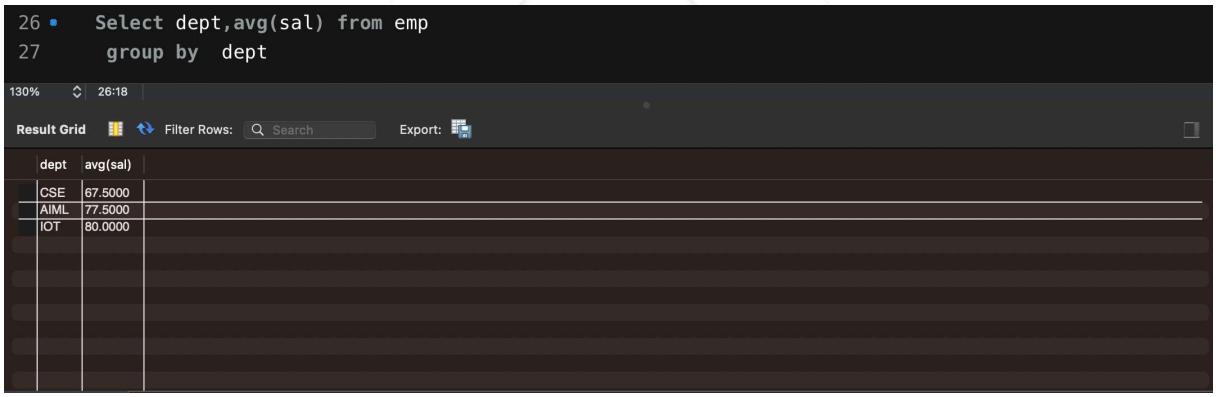
- COUNT
- SUM
- AVG
- MIN
- MAX
- GROUP BY

CODE :

(I) Find dept wise avg salary

```
Select dept, avg(sal) from emp group by dept;
```

OUTPUT :



(ii). Find number of employees in each dept.

CODE :

```
SELECT dept, COUNT(*)  FROM emp WHERE dept IN ('AIML', 'CSE','IOT') GROUP BY Dept;
```

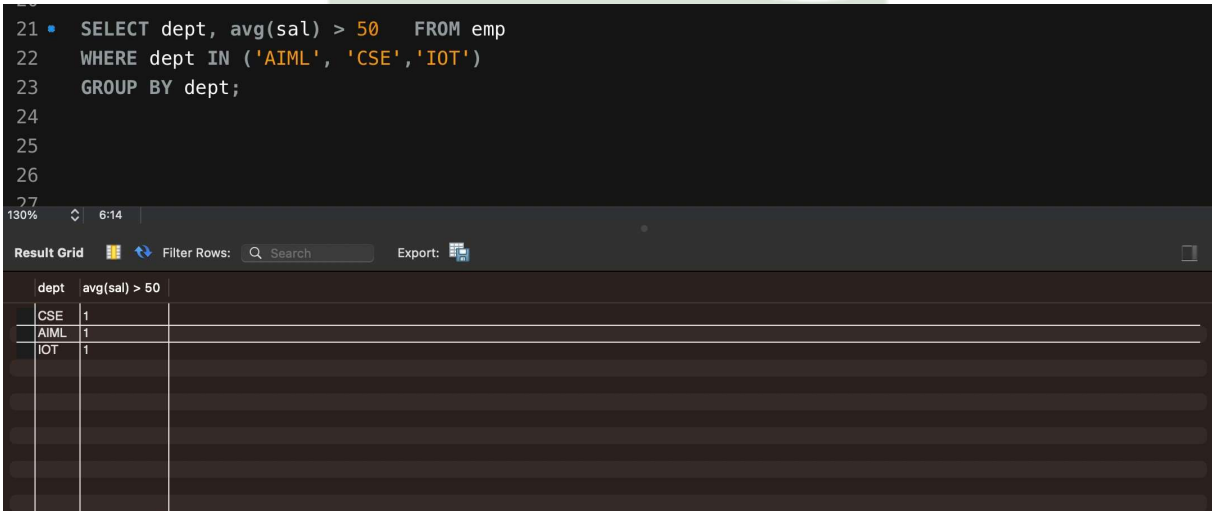
OUTPUT :

(iii). Find depts whose average salary is less than 50k

CODE :

```
SELECT dept, avg(sal) > 50   FROM emp
WHERE dept IN ('AIML', 'CSE','IOT')
GROUP BY dept;
```

OUTPUT :



RESULT :- The concept of aggregation functions was studied and its implementation was extensively covered in MySQL.

****

EXPERIMENT 6

CONSTRAINTS

AIM : (i) Create a table with not null, unique, primary key and check constraints. Write SQL commands for both violation and Non violation of all constraints.

(ii) Create two tables emp and dept where did is common column with Integrity Referential constraint. Illustrate delete and insert rules associated with constraint using SQL commands.

DESCRIPTION : Constraints in SQL are rules and conditions applied to the columns of a table to maintain data integrity and enforce certain rules on the data stored in the database. They help ensure that the data in the database is accurate, consistent, and meets the desired business rules. SQL provides various types of constraints that can be applied to columns:

- NOT NULL
- PRIMARY KEY
- FOREIGN KEY
- UNIQUE
- CHECK

CODE :

(i) Create a table with not null, unique, primary key and check constraints. Write SQL commands for both violation and Non violation of all constraints.

```
CREATE TABLE EMP (emp_id INT PRIMARY KEY,
    emp_salary DECIMAL(10, 2),
    emp_name VARCHAR(50) NOT NULL,
    emp_email VARCHAR(100) NOT NULL UNIQUE,
    emp_age INT CHECK (emp_age >= 18));
```

Violation of the null constraint

```
INSERT INTO EMP (emp_id, emp_name, emp_email, emp_salary,emp_age)
VALUES (3, NULL, 'arun103@gmail.com', 40000, 37);
```

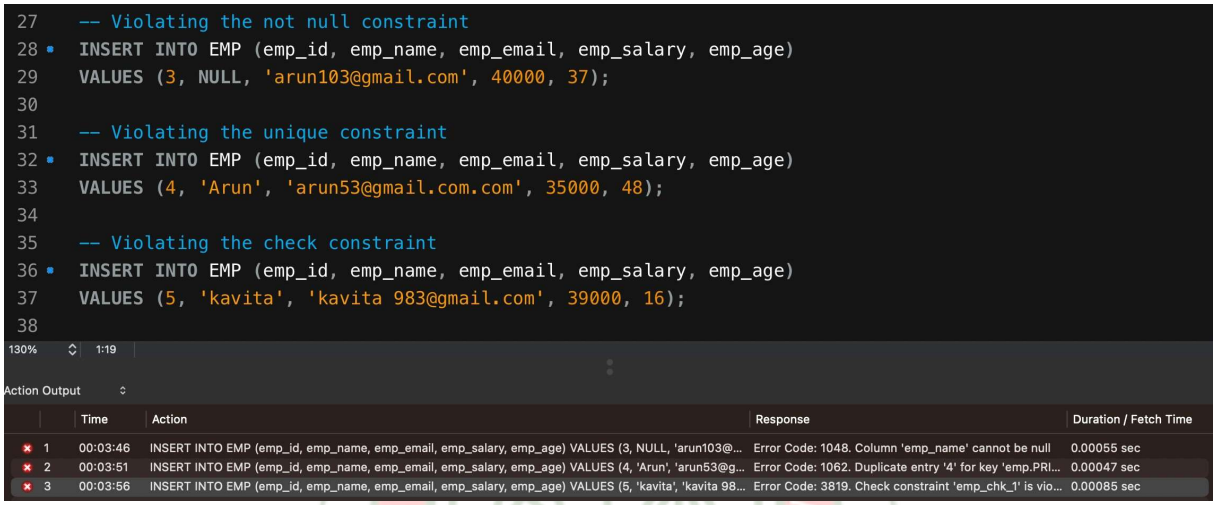
Violation of the unique constraint

```
INSERT INTO EMP (emp_id, emp_name, emp_email, emp_salary, emp_age)
VALUES (4, 'Arun', 'jyothi53@gmail.com.com', 35000, 48);
```

Violation of the check constraint

```
INSERT INTO EMP (emp_id, emp_name, emp_email, emp_salary, emp_age)
VALUES (5, 'kavita', 'kavita 983@gmail.com', 39000, 16);
```

OUTPUT :



```
27   -- Violating the not null constraint
28 • INSERT INTO EMP (emp_id, emp_name, emp_email, emp_salary, emp_age)
29   VALUES (3, NULL, 'arun103@gmail.com', 40000, 37);
30
31   -- Violating the unique constraint
32 • INSERT INTO EMP (emp_id, emp_name, emp_email, emp_salary, emp_age)
33   VALUES (4, 'Arun', 'arun53@gmail.com.com', 35000, 48);
34
35   -- Violating the check constraint
36 • INSERT INTO EMP (emp_id, emp_name, emp_email, emp_salary, emp_age)
37   VALUES (5, 'kavita', 'kavita 983@gmail.com', 39000, 16);
38
```

130% 1:19

Action Output

	Time	Action	Response	Duration / Fetch Time
1	00:03:46	INSERT INTO EMP (emp_id, emp_name, emp_email, emp_salary, emp_age) VALUES (3, NULL, 'arun103@...	Error Code: 1048. Column 'emp_name' cannot be null	0.00055 sec
2	00:03:51	INSERT INTO EMP (emp_id, emp_name, emp_email, emp_salary, emp_age) VALUES (4, 'Arun', 'arun53@g...	Error Code: 1062. Duplicate entry '4' for key 'emp.PRI...	0.00047 sec
3	00:03:56	INSERT INTO EMP (emp_id, emp_name, emp_email, emp_salary, emp_age) VALUES (5, 'kavita', 'kavita 98...	Error Code: 3819. Check constraint 'emp_chk_1' is vio...	0.00085 sec

Non-violation of the the constraints

```
INSERT INTO EMP (emp_id, emp_name, emp_email, emp_salary, emp_age)
VALUES (101, 'RAM', 'ram312@gmailcom', 50000, 30);
```

```
INSERT INTO EMP (emp_id, emp_name, emp_email, emp_salary, emp_age)
VALUES (102, 'KIRAN', 'kiran123@gmail.com', 60000, 28);
```

```
INSERT INTO EMP (emp_id, emp_name, emp_email, emp_salary, emp_age)
VALUES (103, 'ARUN', 'arun103@gmail.com', 40000, 37);
```

```
INSERT INTO EMP (emp_id, emp_name, emp_email, emp_salary, emp_age)
VALUES (104, 'JYOTHI', 'jyothi53@gmail.com', 35000, 48);
```

```
INSERT INTO EMP (emp_id, emp_name, emp_email, emp_salary, emp_age)
VALUES (105, 'KAVITA', 'kavita983@gmail.com', 39000, 22);
```

OUTPUT :

(ii) Create two tables emp and dept where did is common column with Integrity Referential constraint. Illustrate delete and insert rules associated with constraint using SQL commands.

```
-- Inserting data that does not violate the constraints
INSERT INTO EMP (emp_id, emp_name, emp_email, emp_salary, emp_age)
VALUES (101, 'RAM', 'ram312@gmailcom', 50000, 30);

INSERT INTO EMP (emp_id, emp_name, emp_email, emp_salary, emp_age)
VALUES (102, 'KIRAN', 'kiran123@gmail.com', 60000, 28);

INSERT INTO EMP (emp_id, emp_name, emp_email, emp_salary, emp_age)
VALUES (103, 'ARUN', 'arun103@gmail.com', 40000, 37);

INSERT INTO EMP (emp_id, emp_name, emp_email, emp_salary, emp_age)
VALUES (104, 'JYOTHI', 'jyothi53@gmail.com', 35000, 48);

INSERT INTO EMP (emp_id, emp_name, emp_email, emp_salary, emp_age)
VALUES (105, 'KAVITA', 'kavita983@gmail.com', 39000, 22);
```

3:8

Result Grid | Filter Rows: Search | Edit: | Export/Import:

emp_id	emp_name	emp_email	emp_salary	emp_age
101	RAM	ram312@gmailcom	50000.00	30
102	KIRAN	kiran123@gmail.com	60000.00	28
103	ARUN	arun103@gmail.com	40000.00	37
104	JYOTHI	jyothi53@gmail.com	35000.00	48
105	KAVITA	kavita983@gmail.com	39000.00	22

Creating Dept table :

CREATE TABLE dept (did INT PRIMARY KEY, department_name VARCHAR(50) NOT NUL);

Create the emp table with a foreign key constraint referencing dept table :-

CREATE TABLE emp (eid INT PRIMARY KEY, employee_name VARCHAR(50) NOT NULL, did INT, CONSTRAINT fk_emp_dept FOREIGN KEY (did) REFERENCES dept (did) ON DELETE CASCADE ON UPDATE CASCADE);

```
CREATE TABLE dept (
    did INT PRIMARY KEY,
    department_name VARCHAR(50) NOT NULL
);

CREATE TABLE emp (
    eid INT PRIMARY KEY,
    employee_name VARCHAR(50) NOT NULL,
    did INT,
    CONSTRAINT fk_emp_dept FOREIGN KEY (did) REFERENCES dept (did)
        ON DELETE CASCADE
        ON UPDATE CASCADE
);
```

```
INSERT INTO dept (did, department_name) VALUES
(1, 'CSE'),
(2, 'AIML'),
(3, 'IOT');

-- This will succeed because did=1 exists in dept table
INSERT INTO emp (eid, employee_name, did) VALUES (101, 'John Doe', 1);

-- This will fail because did=4 does not exist in dept table
INSERT INTO emp (eid, employee_name, did) VALUES (102, 'Jane Smith', 4);
-- Output: ERROR: insert or update on table "emp" violates foreign key constraint "fk_emp_dept"
-- DETAIL: Key (did)=(4) is not present in table "dept".
```

RESULT : The concept of Constraints was studied and its implementation was extensively covered in MySQL.

****

EXPERIMENT 7

SUBQUERIES

AIM : Use Two tables (emp and dept – common column is did- note dname is in dept only)

(i). Find emp names who are working for CSE and AIML.

(ii). Find emp names who are having salary greater than avg salary of CSE Dept.

(iii). Find the name of employee who is having maximum salary in AIML Dept.

(iv). Find dept name which is having maximum average salary.

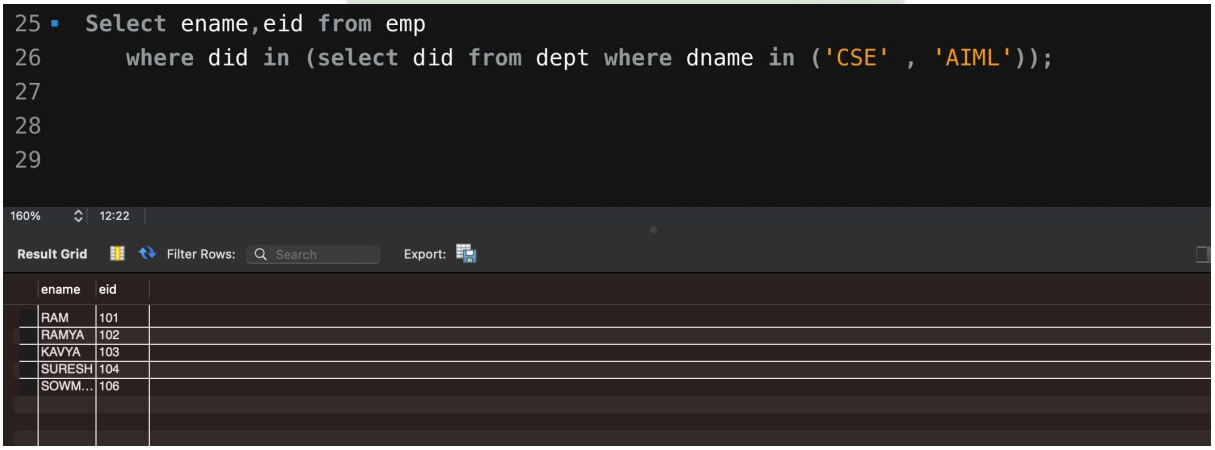
DESCRIPTION : A subquery, also known as a nested query or inner query, is a query that is embedded within another query. It allows you to retrieve data from one or more tables based on the results of a separate query. The result of the inner query is then used in the outer query to perform further operations or filtering.

CODE :

(i) Find emp names who are working for CSE and AIML.

```
Select ename, eid from emp where did in (Select did from dept where dname in ('CSE' , 'AIML'));
```

OUTPUT :



(ii) Find emp names who are having salary greater than avg salary of CSE Dept.

```
Select ename ,eid from emp
where sal >   (Select avg(sal) from emp where did =
(Select did from dept where dname ='CSE'));
```

OUTPUT :

```
30 •  Select ename ,eid from emp
31      where sal >
32      (Select avg(sal) from emp where did =
33      (Select did from dept where dname ='CSE'));
```

160% 6:26

Result Grid Filter Rows: Search Export:

ename	eid
KAVYA	103
SURESH	104
RAHUL	105
SOWMYA	106

(iii) Find the name of employee who is having maximum salary in AIML Dept

```
Select ename ,eid from emp
where sal = (Select max(sal) from emp where did =
(Select did from dept where dname = 'AIML'));
```

OUTPUT :

```
30 •  Select ename ,eid from emp
31      where sal = (Select max(sal) from emp where did =
32      (Select did from dept where dname = 'AIML'));
33
34
```

160% 40:25

Result Grid Filter Rows: Search Export:

ename
RAM

(iv) Find dept name which is having maximum average salary.

```
Select dname from dept
Where did = (Select did from emp order by sal desc limit 1);
```

OUTPUT :

```
30 •  SELECT dname FROM dept
31     WHERE did = (SELECT did FROM emp ORDER BY sal DESC LIMIT 1);
32
33
34
```

160% 19:28

Result Grid Filter Rows: Search Export:

dname
AIML

RESULT : The concept of Sub Queries was studied and its implementation was extensively covered in MySQL.

****

EXPERIMENT 8

JOIN OPERATIONS

AIM : Using the two tables(dept and emp) and the join operations retrieve the data for following queries.

1. Find enames and their concern dnames.
2. Find enames and their concern dnames and also select enames who are not working for any dept from dept table.
3. Find enames and their concern dnames and also select dnames in which no emplyees are working.

DESCRIPTION : In the context of databases and SQL (Structured Query Language), join operations are used to combine data from two or more database tables based on a related column or condition. Join operations allow you to retrieve data from multiple tables simultaneously, creating a result set that contains information from both tables. Some of the most commonly used join operations are as follows :-

i. Inner Join

ii. Left Join

iii. Right Join

iv. Full Join

v. Self Join

vi. Cross Join.

CODE :

1. Find enames and their concern dnames.

```
Select ename ,dname from emp inner join dept on emp.did = dept.did;
```

OUTPUT:

```
24 •  Select ename ,dname
25    from emp inner join dept on emp.did = dept.did;
26
27
```

160% 47:18

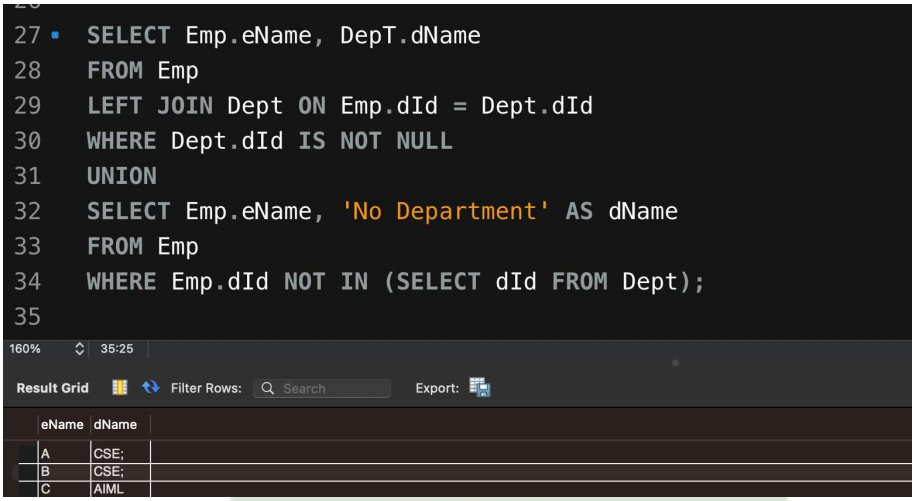
Result Grid Filter Rows: Search Export:

ename	dname
A	CSE
B	CSE
C	AIML
D	AIML
E	CSE
F	AIML

2. Find enames and their concern dnames and also select enames who are not working for any dept from dept table.

```
SELECT Emp.eName, DepT.dName FROM Emp

LEFT JOIN Dept ON Emp.dId = Dept.dId WHERE Dept.dId IS NOT

NULL UNION  SELECT Emp.eName, 'No Department' AS dNameFROM

Emp WHERE Emp.dId NOT IN (SELECT dId FROM Dept);
```

OUTPUT :



3. Find enames and their concern dnames and also select dnames in which no emplyees are working.

```
SELECT Emp.eName, Dept.dName
FROM Emp
LEFT JOIN Dept ON Emp.dId = Dept.dId
UNION
SELECT NULL  AS eName, Dept.dName
FROM Dept
WHERE Dept.dId NOT IN (SELECT dId FROM Emp);
```

OUTPUT :

```
39 • SELECT Emp.eName, Dept.dName
40   FROM Emp
41   LEFT JOIN Dept ON Emp.dId = Dept.dId
42   UNION
43   SELECT NULL AS eName, Dept.dName
44   FROM Dept
45   WHERE Dept.dId NOT IN (SELECT dId FROM Emp);
46
47
48
```

160% 1:37

Result Grid Filter Rows: Search Export:

eName	dName
A	CSE;
B	CSE;
C	AIML
D	AIML

RESULT : The concept of Join Operations was studied and its implementation was extensively covered in MySQL.

****

EXPERIMENT 9

NESTED SUBQUERIES

AIM : Using the two tables (emp and dept) and the concept of Nested subqueries retrieve the data for the following queries.

1. Find dname which is having max avg sal.
2. Find 8th highest salary and its dname.

DESCRIPTION : Nested subqueries, also known as nested queries or subqueries within subqueries, are a powerful feature in SQL that allow you to use the result of one query as a component within another query. A nested subquery is a SELECT statement that is embedded within another SELECT, INSERT, UPDATE, or DELETE statement.

CODE :

1. Find dname which is having max avg sal.

SELECT dname FROM dept WHERE did =
(SELECT did FROM emp GROUP BY did HAVING AVG(sal) = (SELECT
MAX(avg_sal)FROM SELECT AVG(sal) AS avg_sal FROM emp GROUP BY did) As subquery));

OUTPUT :

```
50 •  SELECT dname
51    FROM dept
52    WHERE did = (
53      SELECT did
54      FROM emp
55      GROUP BY did
56      HAVING AVG(sal) = (
57        SELECT MAX(avg_sal)
58        FROM (
59          SELECT AVG(sal) AS avg_sal
60          FROM emp
61          GROUP BY did
62        ) AS subquery
63    )
```

130% 1:49

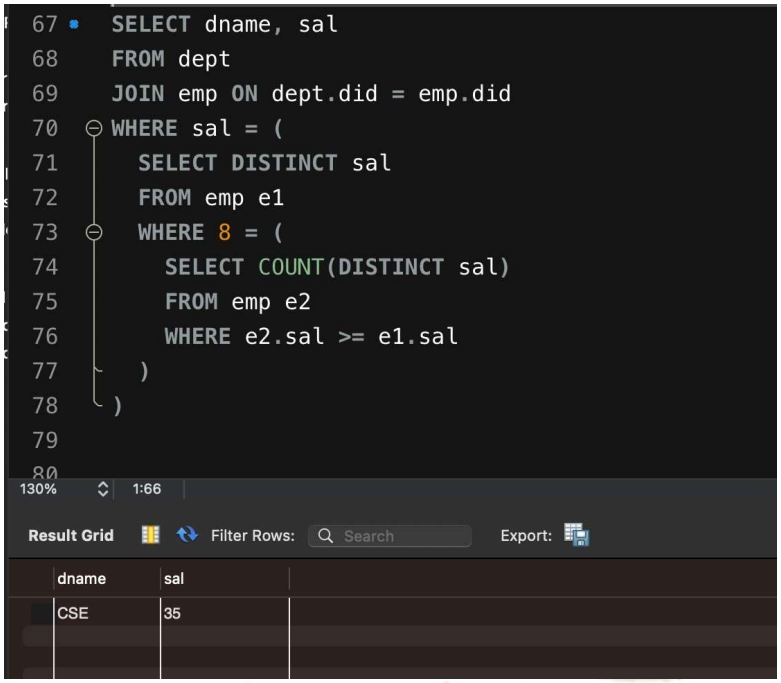
Result Grid Filter Rows: Search Export:

dname
AIML

2. Find 8th highest salary and its dname

```
SELECT dname, sal FROM dept
    JOIN emp ON dept.did = emp.did WHERE sal = (
        SELECT DISTINCT sal  FROM emp e1 WHERE 8 = (
            SELECT COUNT(DISTINCT sal) FROM emp e2
                WHERE e2.sal >= e1.sal));
```

OUTPUT :



RESULT :- Hence executed the above queries using the concept of Nested Subqueries.

****

EXPERIMENT 10

CORELATION SUBQUERIES

AIM : Using the concept of Correlation subqueries retrieve the data for the following queries.

1. Find enames who are having sal > avg sal of their own dept.
2. Find records which are repeated more than once.
3. Find records which are seen exactly once in table.

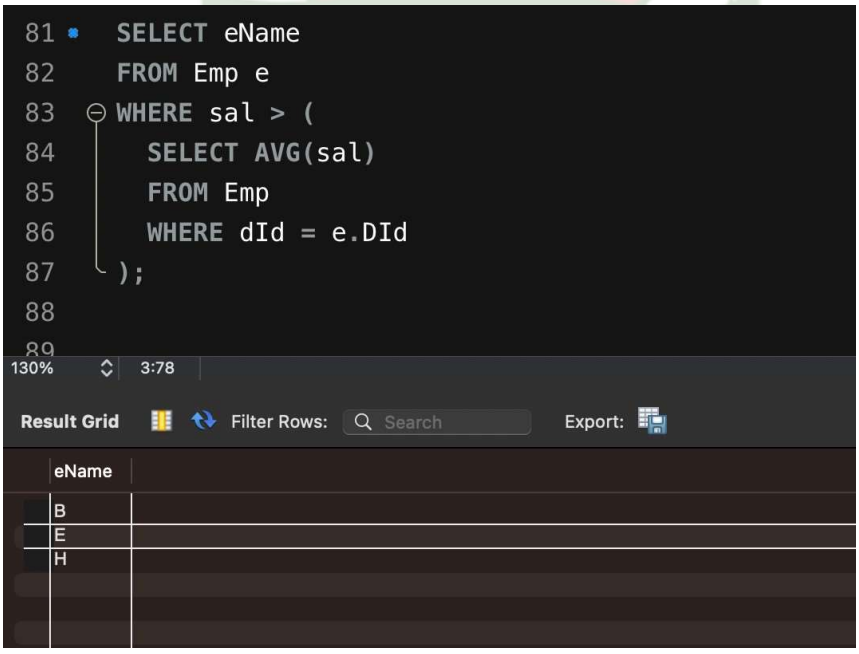
DESCRIPTION : Correlated subqueries, also known as correlated sub-selects or correlated subqueries, are a type of subquery in SQL that depend on the outer query for their results. Unlike regular subqueries, which are independent and executed once, correlated subqueries are executed for each row of the outer query. They allow you to establish a relationship between the inner and outer queries, enabling more complex and context-dependent data retrieval.

CODE :

1.Find enames who are having sal > avg sal of their own dept

```
SELECT ename FROM Emp e WHERE sal >
    (SELECT AVG(sal) FROM     Emp WHERE dId = e.DId);
```

OUTPUT :



2. Find records which are repeated more than once.

OUTPUT :

```
SELECT e_id, e.ename
FROM emp e
WHERE (
  SELECT COUNT(*)
  FROM emp
  WHERE e_id = e.e_id
) > 1;
```

130% 3:87

Result Grid Filter Rows: Search Export:

e_id	ename
107	G
107	G

3. Find records which are seen exactly once in the table.

SELECT emp.e_id, emp.ename FROM emp WHERE (SELECT COUNT(*)
FROM emp AS inner_emp WHERE emp.e_id = inner_emp.e_id) = 1;

OUTPUT:

```
SELECT emp.e_id, emp.ename
FROM emp
WHERE (
  SELECT COUNT(*)
  FROM emp AS inner_emp
  WHERE emp.e_id = inner_emp.e_id
) = 1;
```

130% 1:96

Result Grid Filter Rows: Search Export:

e_id	ename
101	A
102	B
103	C
104	D
105	E
106	F

RESULT :- Hence executed the above queries using the concept of Correlation Subqueries.

****

EXPERIMENT 11

PL SQL

AIM : PLSQL

1. Program on Cursor to print the names and salaries of the employees
2. Function to calculate the total salary of employees in a department:
3. Procedure to update the salary of an employee:
4. Trigger to update the last modification date when a new employee is inserted

DESCRIPTION

PL SQL is a combination of SQL along with procedural features of programming languages. It was developed by oracle corporation in early 90's to enhance the capabilities of SQL.PL SQL is one of three key programming languages embedded in the oracle database along with SQL itself and java

CURSOR: A cursor is a pointer to this context area. PL/SQL controls the context area through a cursor. A cursor holds the rows (one or more) returned by a SQL statement. The set of rows the cursor holds is referred to as the active set.

FUNCTIONS: A standalone function is created using the CREATE FUNCTION statement. While creating a function, you give a definition of what the function has to do. To use a function, you will have to call that function to perform the defined task. When a program calls a function, the program control is transferred to the called function. A called function performs the defined task and when its return statement is executed or when the last end statement is reached, it returns the program control back to the main program.

PROCEDURES: PL/SQL subprogram is a named PL/SQL block that can be invoked repeatedly. If the subprogram has parameters, their values can differ for each invocation. A subprogram is either a procedure or a function. Typically, you use a procedure to perform an action and a function to compute and return a value.

TRIGGERS : Triggers are stored programs, which are automatically executed or fired when some events occur. Triggers have the following benefits:

1. Generating some derived column values automatically

2. Enforcing Referential Integrity

3. Preventing invalid transactions

4. Synchronous replication of tables

5. Imposing security authorisations.

CODE :

#Program on cursors

CREATE TABLE employees (

```
    emp_id NUMBER PRIMARY KEY,
    emp_name VARCHAR2(100) NOT NULL,
    emp_salary NUMBER
);
INSERT INTO employees (emp_id, emp_name, emp_salary)
VALUES (1, 'John', 50000);
INSERT INTO employees (emp_id, emp_name, emp_salary)
VALUES (2, 'Jane', 60000);
INSERT INTO employees (emp_id, emp_name, emp_salary)
VALUES (3, 'Michael', 55000),
       (4, 'Emily', 70000);
DECLARE
    CURSOR emp_cursor IS
        SELECT emp_name, emp_salary
        FROM employees;

    emp_name employees.emp_name%TYPE;
    emp_salary employees.emp_salary%TYPE;
BEGIN
    OPEN emp_cursor;
    LOOP
        FETCH emp_cursor INTO emp_name, emp_salary;
        EXIT WHEN emp_cursor%NOTFOUND;
        DBMS_OUTPUT.PUT_LINE('Employee Name: ' || emp_name || ', Salary: ' || emp_salary);
    END LOOP;
    CLOSE emp_cursor;
END;
```

#Program on function

```
CREATE TABLE employees (
    emp_id NUMBER PRIMARY KEY,
    emp_name VARCHAR2(100) NOT NULL,
    emp_salary NUMBER,
    department VARCHAR2(50)
);
INSERT INTO employees (emp_id, emp_name, emp_salary, department)
VALUES (1, 'John', 50000, 'IT');

INSERT INTO employees (emp_id, emp_name, emp_salary, department)
VALUES (2, 'Jane', 60000, 'HR');

INSERT INTO employees (emp_id, emp_name, emp_salary, department)
VALUES (3, 'Michael', 55000, 'IT');

INSERT INTO employees (emp_id, emp_name, emp_salary, department)
VALUES (4, 'Emily', 70000, 'Finance');

CREATE OR REPLACE FUNCTION get_total_salary(dept_name VARCHAR2) RETURN NUMBER
IS
    total_salary NUMBER := 0;
BEGIN
    SELECT SUM(emp_salary) INTO total_salary
    FROM employees
    WHERE department = dept_name;

    RETURN total_salary;
END;
```

```
DECLARE
   dept_name VARCHAR2(100) := 'IT';
   total_salary NUMBER;
BEGIN
   total_salary := get_total_salary(dept_name);
   DBMS_OUTPUT.PUT_LINE('Total Salary for ' || dept_name || ': ' || total_salary);
END;
```

#Program on procedures

```
-- Procedure to update the salary of an employee
CREATE OR REPLACE PROCEDURE update_salary(emp_id NUMBER, new_salary NUMBER) IS
BEGIN
   UPDATE employees
   SET emp_salary = new_salary
   WHERE emp_id = emp_id;

   COMMIT;
END;
BEGIN
   update_salary(2, 65000);
   DBMS_OUTPUT.PUT_LINE('Salary updated successfully.');
END;
```

#Program on trigger

```
CREATE TABLE employees (
   emp_id NUMBER PRIMARY KEY,
   emp_name VARCHAR2(100) NOT NULL,
```

```
emp_salary NUMBER,
    department VARCHAR2(50),
    last_modified DATE
);
-- Inserting values into the employees table
INSERT INTO employees (emp_id, emp_name, emp_salary, department)
VALUES (1, 'John', 50000, 'IT');

INSERT INTO employees (emp_id, emp_name, emp_salary, department)
VALUES (2, 'Jane', 60000, 'HR');

INSERT INTO employees (emp_id, emp_name, emp_salary, department)
VALUES (3, 'Michael', 55000, 'IT');

INSERT INTO employees (emp_id, emp_name, emp_salary, department)
VALUES (4, 'Emily', 70000, 'Finance');

CREATE OR REPLACE TRIGGER update_last_modified_date
BEFORE INSERT ON employees
FOR EACH ROW
BEGIN
    :NEW.last_modified := SYSDATE;
END;
INSERT INTO employees (emp_id, emp_name, emp_salary)
VALUES (5, 'Sarah', 60000);
SELECT * FROM employees;
```

OUTPUT:

```
Statement processed.
Employee Name: John, Salary: 50000
Employee Name: Jane, Salary: 60000
Employee Name: Michael, Salary: 55000
Employee Name: Emily, Salary: 70000
```

```
Function created.
```

```
Statement processed.
Total Salary for IT: 105000
```

```
Procedure created.
```

```
Statement processed.
Salary updated successfully.
```

```
Trigger created.
```

```
1 row(s) inserted.
```

EMP_ID	EMP_NAME	EMP_SALARY	DEPARTMENT	LAST_MODIFIED
1	John	50000	IT	-
2	Jane	60000	HR	-
3	Michael	55000	IT	-
4	Emily	70000	Finance	-
5	Sarah	60000	-	04-AUG-23

RESULT : The queries have been successfully executed by using various concepts such as cursors, functions, procedures, triggers.

****

‘

EXPERIMENT 12

LAB INTERNAL 1 – QUESTIONS

AIM : Using the below tables we need to retrieve data from tables for the the given Queries.

GIVEN TABLES :

QUESTION NUMBER : 1

1. customer(cust_id, name, income_permonth, gender, location_pincode) cust_id is pk, location_pincode is fk on city table

2. accounts(account_number, cust_id, account_type, balance_inlakhs, ifsc_code) account_number is pk , cust_id is fk on customers , ifsc_code is fk on branch table

3. branch (ifsc_code, branch_name, location_pincode) ifsc_code is pk

4. city(location_pincode, cityname, state) location_pincode is pk

QUERIES :

1. Find number of accounts in hyderabad city with balance amount > 1 lakh.
2. Find number of branches in vizag city with female accounts > 2.
3. Find customer names from any specified branch like Gandipet who are having more than one account. (like savings, current , loan etc).
4. Find city name which is having minimum average balance amount among all cities.

CODE :

1. Find number of accounts in hyderabad city with balance amount > 1 lakh

```
SELECT COUNT(*) AS num_accounts FROM accounts a WHERE a.cust_id IN (

    SELECT c.cust_id FROM customer c

    JOIN branch b ON c.location_pincode = b.location_pincode

    JOIN city ct ON b.location_pincode = ct.location_pincode  WHERE ct.cityname = 'Hyderabad')

    AND a.balance_inlakhs > 1;
```

OUTPUT :

```
54 •   SELECT COUNT(*) AS num_accounts
55    FROM accounts a
56    WHERE a.cust_id IN (
57        SELECT c.cust_id
58        FROM customer c
59        JOIN branch b ON c.location_pincode = b.location_pincode
60        JOIN city ct ON b.location_pincode = ct.location_pincode
61        WHERE ct.cityname = 'Hyderabad'
62    )
63    AND a.balance_inlakhs > 1;
```

160% 21:56

Result Grid Filter Rows: Search Export:

num_accounts
1

2. Find number of branches in vizag city with female accounts > 2

```
SELECT COUNT(DISTINCT b.branch_name) AS num_branches
FROM branch b
JOIN city cty ON b.location_pincode = cty.location_pincode
JOIN customer c ON b.location_pincode = c.location_pincode
WHERE cty.cityname = 'Vizag'
AND c.gender = 'Female'
GROUP BY b.branch_name
HAVING COUNT(DISTINCT c.cust_id) > 2;
```

OUTPUT :

```
66 •  SELECT COUNT(DISTINCT b.branch_name) AS num_branches
67    FROM branch b
68    JOIN city cty ON b.location_pincode = cty.location_pincode
69    JOIN customer c ON b.location_pincode = c.location_pincode
70    WHERE cty.cityname = 'Vizag'
71      AND c.gender = 'Female'
72    GROUP BY b.branch_name
73    HAVING COUNT(DISTINCT c.cust_id) > 2;
74
```

160% 27:63

Result Grid Filter Rows: Search Export:

num_branches
1

3. Find customer names from any specified branch like gandipet who are having more than one account. (like savings, current , loan etc).

SELECT c.name FROM customer c WHERE c.location_pincode

IN (SELECT b.location_pincode FROM branch b WHERE b.branch_name = ‘Gandipet’) AND c.cust_id

IN (SELECT cust_id FROM accounts GROUP BY cust_id HAVING COUNT(*) > 1);

OUTPUT :

```
80 •  SELECT c.name
81    FROM customer c
82    WHERE c.location_pincode IN (
83        SELECT b.location_pincode
84        FROM branch b
85        WHERE b.branch_name = 'Gandipet'
86    )
87    AND c.cust_id IN (
88        SELECT cust_id
89        FROM accounts
90        GROUP BY cust_id
91        HAVING COUNT(*) > 1
92    );
```

160% 18:89

Result Grid Filter Rows: Search Export:

name
ANANYA
RAMYA

4. Find city name which is having minimum average balance amount among all cities.

SELECT cityname, avg_balanceFROM (SELECT ct.cityname, AVG(a.balance_inlakhs)
AS avg_balance FROM accounts a JOIN customer c ON a.cust_id = c.cust_id
JOIN branch b ON c.location_pincode = b.location_pincode
JOIN city ct ON b.location_pincode = ct.location_pincode
GROUP BY ct.cityname) city_avg_balances
ORDER BY avg_balance ASC LIMIT 1;

OUTPUT :

```
96 •  SELECT cityname, avg_balance
97    FROM (
98        SELECT ct.cityname, AVG(a.balance_inlakhs) AS avg_balance
99        FROM accounts a
100       JOIN customer c ON a.cust_id = c.cust_id
101       JOIN branch b ON c.location_pincode = b.location_pincode
102       JOIN city ct ON b.location_pincode = ct.location_pincode
103       GROUP BY ct.cityname
104   ) city_avg_balances
105   ORDER BY avg_balance ASC
106   LIMIT 1;
107
```

160% 62:98

Result Grid | Filter Rows: Search | Export: | Fetch rows:

cityname	avg_balance
Hyderabad	1.250000

QUESTION NUMBER : 2

1. Emp (emp_id, ename, salary, did, eid_of_manager) emp_id is pk and eid of manger is fk on emp_id of same table.

2. dept(did, dname) did is pk.

3. projects(pid, eid)pid is pk and eid is fk on emp.

QUERIES :

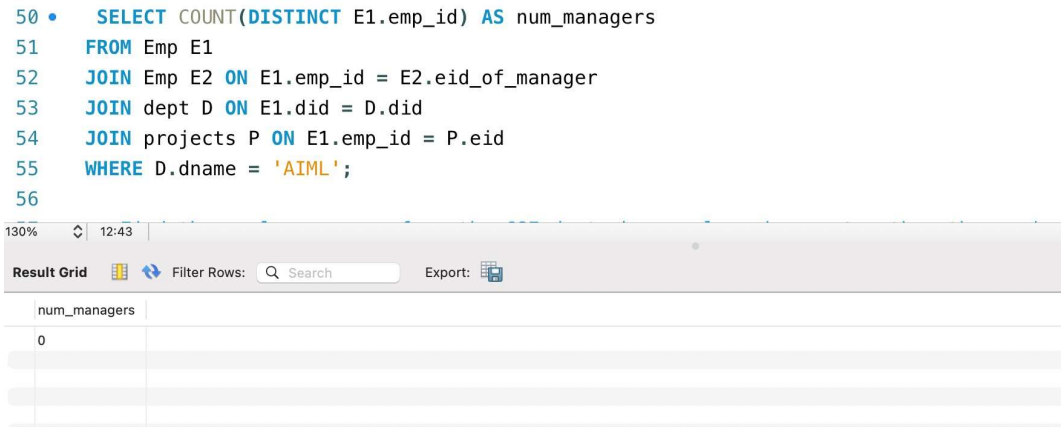
1. Find number of managers from AIML dept who are doing projects
2. Find the emp names from CSE dept whose sal > maximum avg sal of all depts.
3. Find the dept names from which none of the employees are doing atleast one projects
4. Find employee names who are managers to themselves
5. Find employee names who are not managers to any employees.

CODE :

1.Find the number of managers from the AIML dept who are doing projects:

```
SELECT COUNT(DISTINCT E1.emp_id) AS num_managers
FROM Emp E1
JOIN Emp E2 ON E1.emp_id = E2.eid_of_manager
JOIN dept D ON E1.did = D.did
JOIN projects P ON E1.emp_id = P.eid
WHERE D.dname = 'AIML';
```

OUTPUT :



2. Find the employee names from the CSE dept whose salary is greater than the maximum average salary of all departments:

```
SELECT E.ename FROM Emp E JOIN dept D ON E.did = D.did
WHERE D.dname = 'CSE' AND E.salary > (SELECT MAX(avg_salary)
FROM (SELECT AVG(salary) as avg_salary FROM Emp GROUP BY did) AS avg_salaries);
```

OUTPUT :

```
59 •  SELECT E.ename
60    FROM Emp E
61    JOIN dept D ON E.did = D.did
62    WHERE D.dname = 'CSE' AND E.salary > (
63        SELECT MAX(avg_salary)
64        FROM (
65            SELECT AVG(salary) as avg_salary
66            FROM Emp
67            GROUP BY did
68        ) AS avg_salaries
69    );
70
71    -- Find the department names from which none of the employees are doing at least one project:
```

130% 24:55

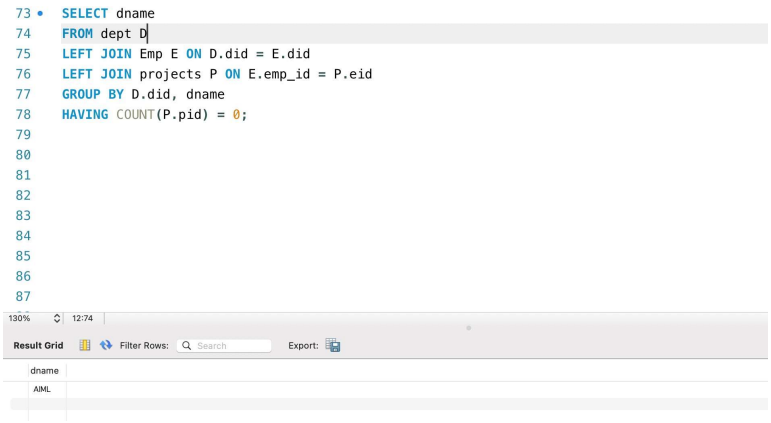
Result Grid Filter Rows: Search Export:

ename
Harshit

3. Find the dept names from which none of the employees are doing atleast one projects:

```
SELECT dname FROM dept D LEFT JOIN Emp E ON D.did = E.did
LEFT JOIN projects P ON E.emp_id = P.eid  GROUP BY D.did, dname
HAVING COUNT(P.pid) = 0;
```

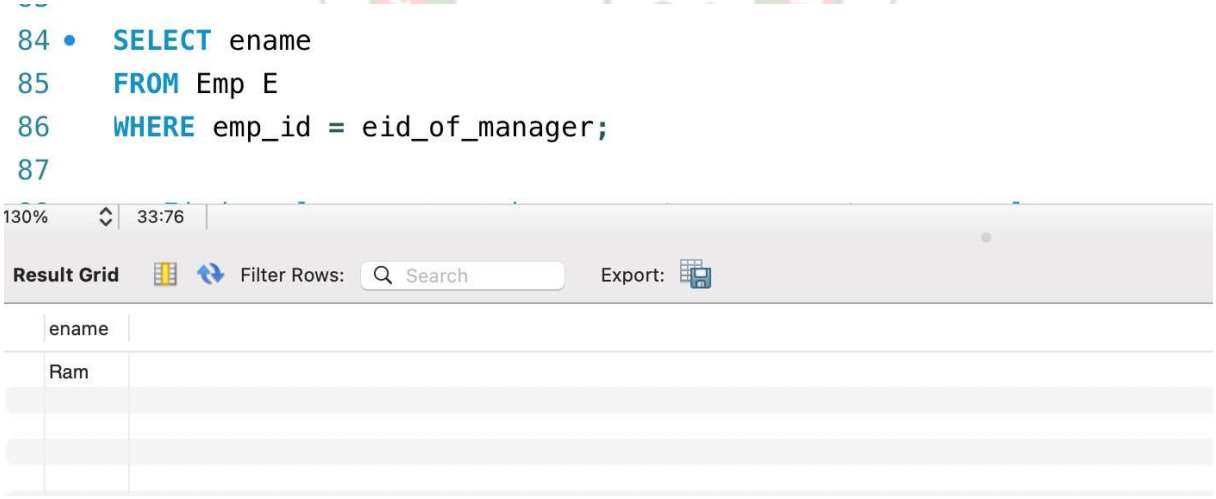
OUTPUT :



4. Find employee names who are managers to themselves.

Select Ename from Emp E where emp_id = eid_of _managers ;

OUTPUT :



5. Find employee names who are not managers to any employees.

SELECT ename FROM Emp E
WHERE emp_id NOT IN (SELECT DISTINCT eid_of_manager FROM Emp);

OUTPUT :

```
89 •  SELECT ename
90    FROM Emp E
91    WHERE emp_id NOT IN (SELECT DISTINCT eid_of_manager FROM Emp);
92
93
94
95
96
```

130% 13:84

Result Grid Filter Rows: Search Export:

ename
Rocky
Oberoi

QUESTION NUMBER : 3

TABLES :

1. artists (artist_id, name, location_pincode) artist_id is pk, location pincode is fk on city table.
2. skills (skills_id, skill_name)skill id is pk.
3. artists_skills(artist_id, skill_id, exp_in_years)artist_id and skill_id combined pk ie composite pk and skill_id is fk on skills table.
4. movies(movie_name,artist_id, language, genre)movie_name and artist_id are composite pk , genres like action, horror etc.
5. city(pincode, cityname, state)pincode is pk.

QUERIES :

1. Find the artists who are both directors and producers for action genre.
2. Find music directors from hyderabad city who have at least 5 years of experience as singers in telugu movies.
3. Find all artists names who commonly worked for kgf1, RRR and pushpa movies.

CODE :

1. Find the artists who are both directors and producers for action genre.

```
SELECT name FROM artists WHERE artist_id IN (SELECT artist_id  FROM artists_skills

WHERE skill_id IN (SELECT skill_id FROM skills  WHERE skill_name IN
```

('Director', 'Producer')) GROUP BY artist_id HAVING COUNT(DISTINCT skill_id) = 2)

AND artist_id IN (SELECT artist_id FROM moviesWHERE genre = ‘Action');

OUTPUT :

```
76 •   SELECT name
77     FROM artists
78   WHERE artist_id IN (
79     SELECT artist_id
80     FROM artists_skills
81     WHERE skill_id IN (
82       SELECT skill_id
83       FROM skills
84       WHERE skill_name IN ('Director', 'Producer')
85     )
86     GROUP BY artist_id
87     HAVING COUNT(DISTINCT skill_id) = 2
88   ) AND artist_id IN (
89     SELECT artist_id
90     FROM movies
91     WHERE genre = 'Action'
92   );
```

130% 3:92

Result Grid Filter Rows: Search Export:

name
Allu arjun

2. Find music directors from Hyderabad city who have at least 5 years of experience as singers in telugu movies.

SELECT name FROM artists WHERE artist_id IN (SELECT artist_id

FROM artists_skills WHERE skill_id =

(SELECT skill_id FROM skills WHERE skill_name = 'Music Director'))

AND artist_id IN (SELECT artist_id FROM artists_skills WHERE skill_id =

(SELECT skill_id FROM skills WHERE skill_name = 'Singer')

AND exp_in_years >= 5) AND location_pincode

IN (SELECT pincode FROM city WHERE cityname = ‘Hyderabad');

OUTPUT :

```
97 •  SELECT name FROM artists WHERE artist_id IN (SELECT artist_id
98    FROM artists_skills WHERE skill_id = (SELECT skill_id FROM skills WHERE skill_name = 'Music Director'))
99    AND artist_id IN (SELECT artist_id FROM artists_skills WHERE skill_id =
00    (SELECT skill_id FROM skills WHERE skill_name = 'Singer')
01      AND exp_in_years >= 5) AND location_pincode
02      IN (SELECT pincode FROM city WHERE cityname = 'Hyderabad');
03
04
05
06    -- Find all artists' names who commonly worked for KGF1, RRR, and Pushpa movies:
07
08 •  SELECT name
09    FROM artists
```

30% 1:94

Result Grid | Filter Rows: Search | Export:

name
NTR

3. Find all artists names who commonly worked for kgf1, RRR and pushpa movies.

SELECT name FROM artists WHERE artist_id IN

(SELECT artist_id FROM movies WHERE movie_name IN ('KGF1', 'RRR', 'Pushpa') GROUP BY artist_id HAVING COUNT(DISTINCT movie_name) = 3);

OUTPUT :

```
114 •  SELECT name
115    FROM artists
116    WHERE artist_id IN (
117      SELECT artist_id
118      FROM movies
119      WHERE movie_name IN ( 'KGF1','RRR','Pushpa')
120      GROUP BY artist_id
121      HAVING COUNT(*) = 3
122    );
123
124
```

130% 1:109

Result Grid | Filter Rows: Search | Export:

name
Allu arjun
Sukumar
NTR
Rajamouli

QUESTION NUMBER : 4

TABLES :

1. player(player_id, name, game_account_balance, location_pincode) player_id is pk , location_pincode is fk on city table.
2. matches(match_id, type_of_game , location_pincode).
3. transactions(trans_id, player_id, bet_amount, win_or_loss) Win_or_loss is Boolean column.
4. city(pincode, name) pincode is pk .

QUERIES :

1.Find the player name who lost maximum amount in bets
2. Find city names with maximum average bet amount
3. Find the type of game which is having minimum number of bets
4. find city names from which no citizens bets done so far.

CODE :

1. Find the player name who lost maximum amount in bets.

```
SELECT name FROM player WHERE player_id =

(SELECT player_id FROM transactions WHERE win_or_loss =

FALSE GROUP BY player_id ORDER BY SUM(bet_amount) DESC LIMIT 1);
```

OUTPUT :

```
61 •  SELECT name FROM player WHERE player_id =
62    (SELECT player_id FROM transactions WHERE win_or_loss =
63    FALSE GROUP BY player_id ORDER BY SUM(bet_amount) DESC LIMIT 1);
64
65
```

130% 33:53

Result Grid Filter Rows: Search Export:

name
Jack

2. Find city names with maximum average bet amount

```
SELECT name AS city_name FROM city WHERE pincode =
```

(SELECT location_pincode FROM player WHERE player_id =

(SELECT player_id FROM transactions GROUP BY player_id ORDER BY AVG(bet_amount) DESC LIMIT 1));

OUTPUT :

```
67 •  SELECT name AS city_name FROM city WHERE pincode =
68    (SELECT location_pincode FROM player WHERE player_id =
69     (SELECT player_id FROM transactions GROUP BY player_id ORDER BY AVG(bet_amount) DESC LIMIT 1));
70
```

130% 57:62

Result Grid Filter Rows: Search Export:

city_name
Sydney

3. Find the type of game which is having minimum number of bets.

SELECT type_of_game

FROM matches

WHERE match_id = (SELECT match_id FROM transactions GROUP BY match_id

ORDER BY COUNT(trans_id) LIMIT 1);

OUTPUT :

```
73 •  SELECT type_of_game FROM matches WHERE match_id =
74    (SELECT match_id FROM transactions GROUP BY match_id ORDER BY COUNT(trans_id) LIMIT 1)
75
76
```

130% 1:65

Result Grid Filter Rows: Search Export:

type_of_game
Football
Cricket
Basketball
Tennis

4. Find city names from which no citizens bets done so far.

SELECT name AS city_name FROM city WHERE pincode NOT IN

(SELECT DISTINCT location_pincode FROM player WHERE player_id IN

(SELECT player_id FROM transactions));

OUTPUT :

```
78 •  SELECT name AS city_name FROM city WHERE pincode NOT IN
79    (SELECT DISTINCT location_pincode FROM player WHERE player_id IN
80     (SELECT player_id FROM transactions));
81
82
83
84
85
```

130% 1:77

Result Grid Filter Rows: Search Export:

city_name
Sydney

RESULT : The above queries have been successfully executed by using various concepts such as aggregation functions, sub-queries.

****

EXPERIMENT 13

LAB INTERNAL 2 - QUESTIONS

AIM : Using the below tables we need to retrieve data from tables for the given Queries.

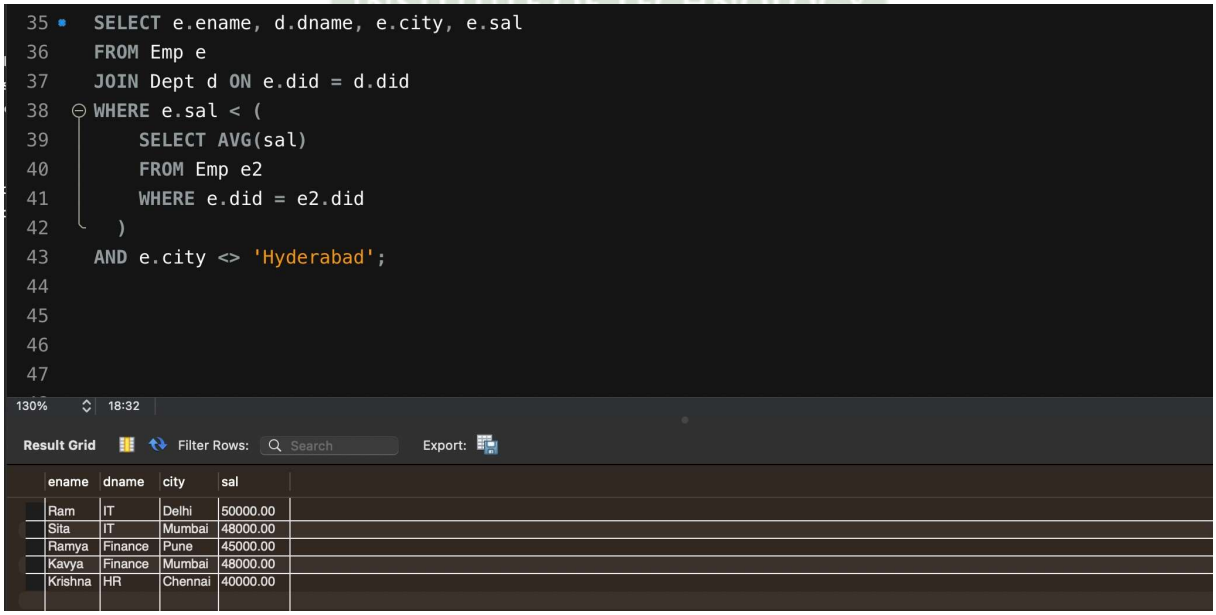
GIVEN TABLES :

1. Consider tables Emp and Dept
Emp (eid, ename, sal, city, did)
Dept (did, dname)
Find enames , concern dname , city, sal of those who are earning less than average salary of their own department and lives in other than Hyderabad

CODE :

```
SELECT e.ename, d.dname, e.city, e.sal

FROM Emp e JOIN Dept d ON e.did = d.did

WHERE e.sal < ( SELECT AVG(sal)

FROM Emp e2   WHERE e.did = e2.did)AND e.city <> 'Hyderabad';
```

OUTPUT:



```
35 • SELECT e.ename, d.dname, e.city, e.sal
36   FROM Emp e
37   JOIN Dept d ON e.did = d.did
38   WHERE e.sal < (
39       SELECT AVG(sal)
40       FROM Emp e2
41       WHERE e.did = e2.did
42     )
43   AND e.city <> 'Hyderabad';
44
45
46
47
```

130% 18:32

Result Grid Filter Rows: Search Export:

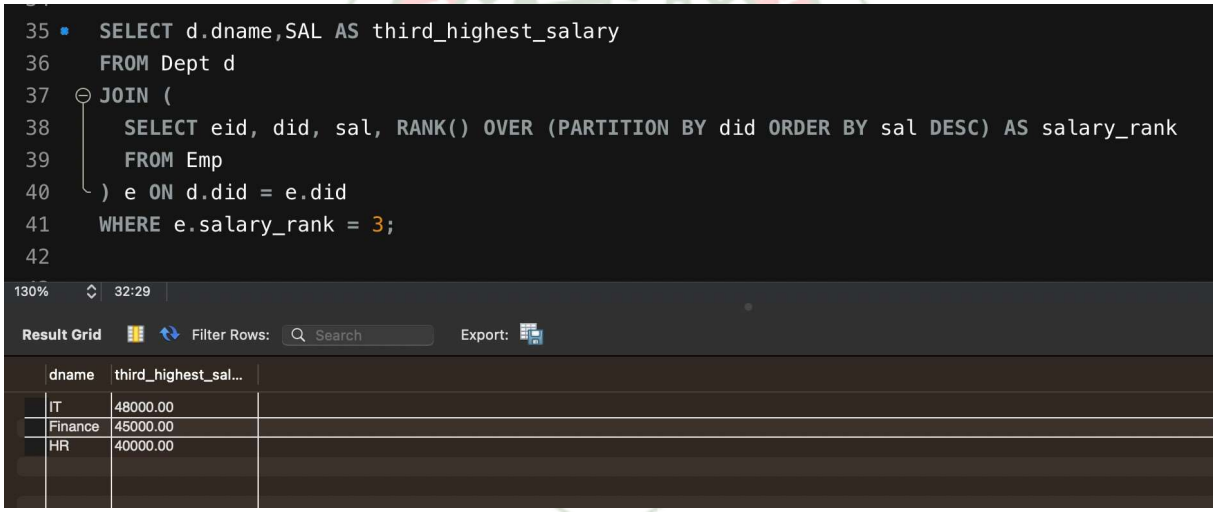
ename	dname	city	sal
Ram	IT	Delhi	50000.00
Sita	IT	Mumbai	48000.00
Ramya	Finance	Pune	45000.00
Kavya	Finance	Mumbai	48000.00
Krishna	HR	Chennai	40000.00

2. Consider tables Emp and Dept.
Emp (eid, ename, sal, did)
Dept (did, dname)
Find 3rd highest salary of each department. Output table should have two columns, one is dname and theother is 3rd highest salary of each department.

CODE :

```
SELECT d.dname,SAL AS third_highest_salary

FROM Dept d JOIN (SELECT eid, did, sal, RANK() OVER

(PARTITION BY did ORDER BY sal DESC) ASsalary_rank FROM Emp) as e

ON d.did =e.didWHERE e.salary_rank = 3;
```

OUTPUT:



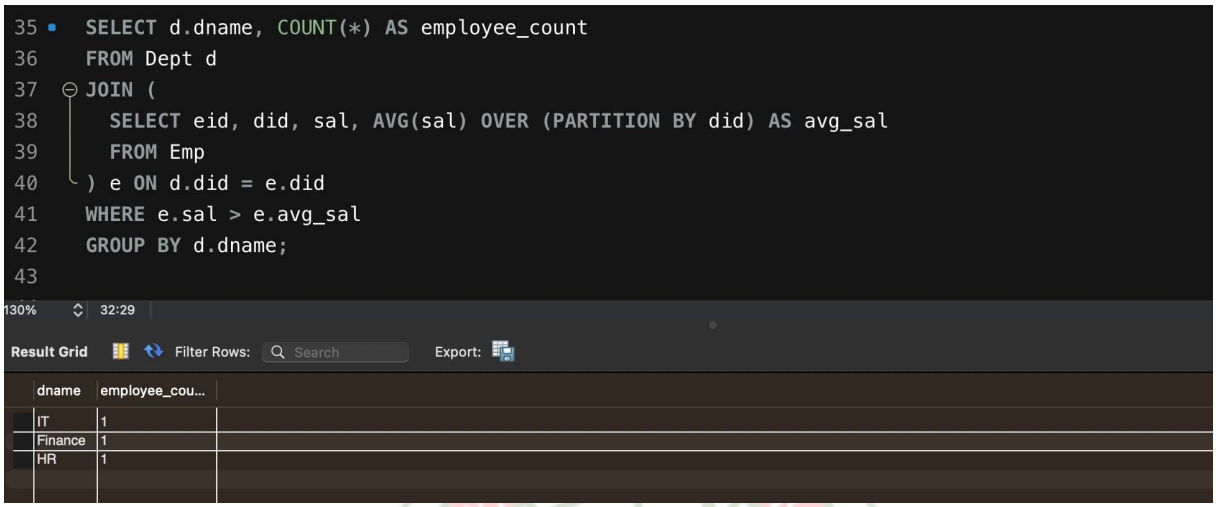
3. Consider tables Emp and Dept.
Emp (eid, ename, sal, did)
Dept (did, dname)
Find the number of employees in each department whose salary is greater than average salary of their own department. Output table should have two column one is dname and other is count of employees as said above.

CODE :-

```
SELECT d.dname, COUNT(*) AS employee_count

FROM Dept d JOIN ( SELECT eid, did, sal, AVG(sal) OVER (PARTITION BY did) AS avg_sal
```

FROM Emp) e ON d.did = e.did WHERE e.sal > e.avg_sal GROUP BY d.dname;

OUTPUT :



4.A. Delete the Unique records (which are seen only once) from table in single shot.

4.B. Delete duplicate records (which are seen more than once) from table in single shot.

Eg :- Before execution table is:

COL1	COL2
X	Y
A	B
X	Y
K	L

AFTER execution of query A:

COL1	COL2
X	Y
X	Y

AFTER execution of query B:

COL1	COL2
A	B
K	L

CODE : 4A

In this table we have take the unique record as 104 and deleted it:

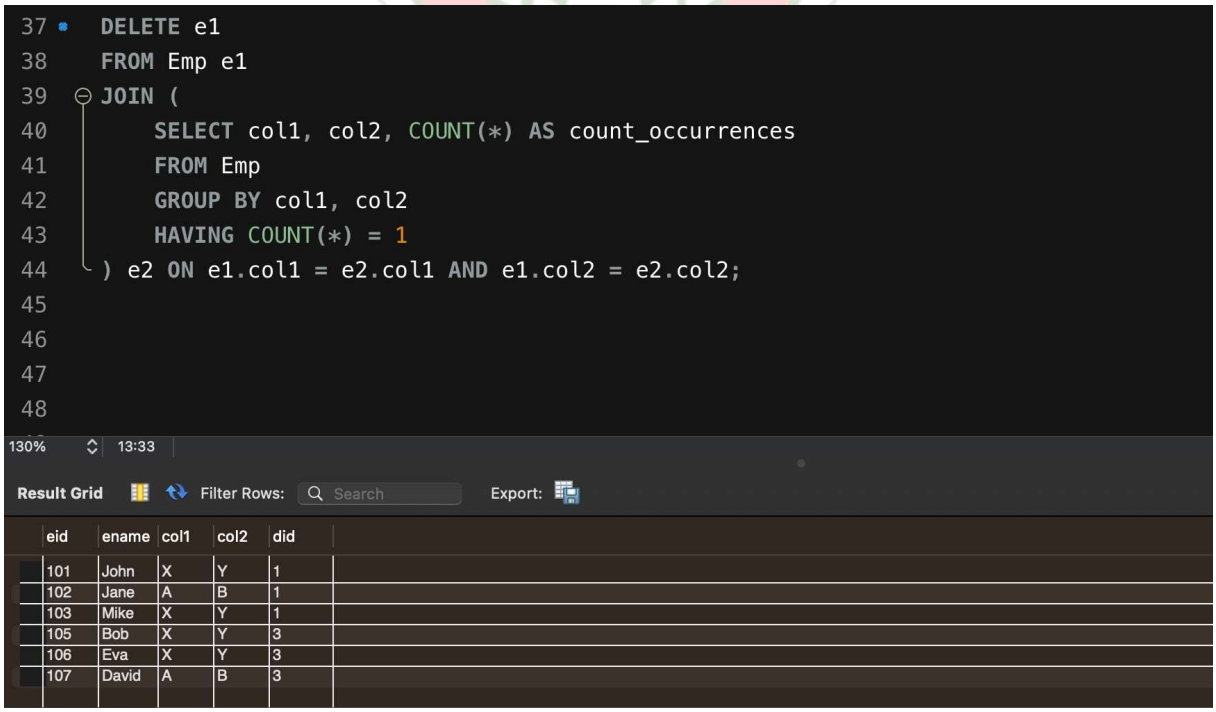
DELETE e1FROM Emp e1

JOIN (SELECT col1, col2, COUNT(*) AS count_occurrences FROM Emp

GROUP BY col1, col2 HAVING COUNT(*) = 1) e2 ON e1.col1 = e2.col1

AND e1.col2 = e2.col2;

OUTPUT :

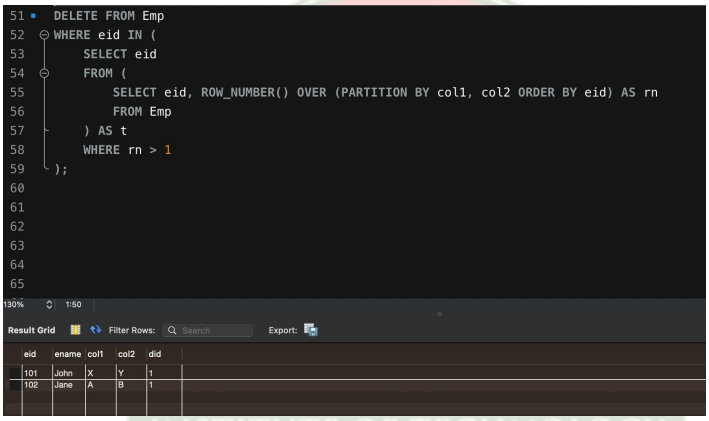


CODE : 4B

In this we have deleted all the remaining duplicate records.

```
DELETE FROM Emp

WHERE eid IN (SELECT eid FROM

(SELECT eid, ROW_NUMBER() OVER

(PARTITION BY col1, col2 ORDER BY eid) AS rn FROM Emp)

AS t WHERE rn > 1);
```

OUTPUT :



RESULT : The queries have been successfully executed by using various concepts such as sub-queries, correlation sub-queries, and join operations.

****