COURSE LABORATORY MANUAL

A. LABORATORY OVERVIEW

Degree:	**BE**	Programme:	CSE
Semester:	5	Academic Year:	2023-24
Laboratory Title:	**DBMS Laboratory with Mini Project**	Laboratory Code:	**21CSL55**
L-T-P-S:	0-0-2-0	Duration of SEE:	3 Hrs
Total Contact Hours:	24	SEE Marks:	50
Credits:	1	CIE Marks:	50
Lab Manual Author:	Mr. Pradeep Kumar KG	Sign	Dt : 15.11.2023
Checked By:	Mr. Krishnamohana A J	Sign	Dt : 15.11.2023

*The SEE will be conducted for 100 marks and proportionally reduced to 60 marks.

B. DESCRIPTION

1. PREREQUISITES:

- Data Structures and Applications (21CS32)
- Programming IN c++ (21CS282)
- Object Oriented Programming with JAVA Laboratory (21CSL35)

2. BASE COURSE:

- Database Management System(21CS53)

3. COURSE OUTCOMES:

At the end of the course, the student will be able to;

CO 1. Create, Update and query on the database.

CO 2. Demonstrate the working of different concepts of DBMS.

CO 3. Implement, analyze and evaluate the project developed for an application.

4. RESOURCES REQUIRED:

- Personal Computer
- Linux distribution OS
- MySQL
- XAMPP, Visual Studio, Netbeans etc. (for project)

Prepared by: Mr. Pradeep Kumar KG   Checked by:Mr. Krishnamohana A J   HOD

5. RELEVANCE OF THE COURSE:

- Web technology and its applications
- Data Mining and Data Warehousing
- Big Data Analytics
- NoSQL Databases

6. GENERAL INSTRUCTIONS:

- Type the following command at the command line terminal in Linux Environment to access the MySQL shell, shell> mysql -u root -p
- Create a new MySQL user account by running the following command:
  mysql>CREATE USER 'newuser'@'localhost' IDENTIFIED BY 'user_password';
- Grant Privileges to a MySQL User Account
- mysql>GRANT ALL PRIVILEGES ON *.* TO 'database_user'@'localhost';
- To disconnect from the MySQL server, type the following command
  mysql> QUIT

7. CONTENTS:

Expt No.	Title of the Experiments	RBT	CO
1	Library Database	L4	CO1, 2
2	Order Database	L4	CO1, 2
3	Movie Database	L4	CO1, 2
4	College Database	L4	CO1, 2
5	Company Database	L4	CO1, 2
6	Open ended experiment - 1		
7	Open ended experiment - 2		

8. REFERENCE:

1. Database systems Models, Languages, Design and Application Programming, Ramez Elmasri and Shamkant B. Navathe, 7th Edition, 2017, Pearson.
2. Database management systems, Ramakrishnan, and Gehrke, 3rd Edition, 2014, McGraw Hill
3. Silberschatz Korth and Sudharshan, Database System Concepts, 6th Edition, Mc-GrawHill, 2013.
4. Coronel, Morris, and Rob, Database Principles Fundamentals of Design, Implementation and Management, Cengage Learning 2012.
5. https://www.w3schools.com/sql
6. https://stackoverflow.com

-

C. EVALUATION SCHEME

Assessment Details (both CIE and SEE)

The weightage of Continuous Internal Evaluation (CIE) is 50% and for Semester End Exam (SEE) is 50%. The minimum passing mark for the CIE is 40% of the maximum marks (20 marks). A student shall be deemed to have satisfied the academic requirements and earned the credits allotted to each course. The student has to secure not less than 35% (18 Marks out of 50) in the semester-end examination (SEE). The student has to secure a minimum of 40% (40 marks out of 100) in the sum total of the CIE (Continuous Internal Evaluation) and SEE (Semester End Examination) taken together.

Continuous Internal Evaluation (CIE):

- CIE marks for the practical course is **50 Marks**.
- The split-up of CIE marks for record/ journal and test are in the ratio **60:40**.
- Each experiment to be evaluated for conduction with an observation sheet and record write-up.
- Record should contain all the specified experiments in the syllabus and each experiment write-up will be evaluated for 10 marks.
- Total marks scored by the students are scaled downed to 30 marks (60% of maximum marks).
- Weightage to be given for neatness and submission of record/write-up on time.
- Department shall conduct 02 tests for 100 marks, the first test shall be conducted after the 8th week of the semester and the second test shall be conducted after the 14th week of the semester.
- In each test, test write-up, conduction of experiment, acceptable result, and procedural knowledge will carry a weightage of 60% and the rest 40% for viva-voce.
- The average of 02 tests is scaled down to 20 marks (40% of the maximum marks).
- The Sum of scaled-down marks scored in the report write-up/journal and average marks of two tests is the total CIE marks scored by the student.

Semester End Evaluation (SEE):

- SEE marks for the practical course is 50 Marks.
- SEE shall be conducted jointly by the two examiners of the same institute, examiners are appointed by the University
- All laboratory experiments are to be included for practical examination.
- (Rubrics) Breakup of marks and the instructions printed on the cover page of the answer script to be strictly adhered to by the examiners. **OR** based on the course requirement evaluation rubrics shall be decided jointly by examiners.
- Students can pick one question (experiment) from the questions lot prepared by the internal /external examiners jointly.

- Evaluation of test write-up/ conduction procedure and result/viva will be conducted jointly by examiners.
- General rubrics suggested for SEE are mentioned here, writeup-20%, Conduction procedure and result in -60%, Viva-voce 20% of maximum marks. SEE for practical shall be evaluated for 100 marks and scored marks shall be scaled down to 50 marks (however, based on course type, rubrics shall be decided by the examiners)
- Students can pick one experiment from the questions lot of PART A with an equal choice to all the students in a batch. For PART B, the project group (Maximum of 4 students per batch) should demonstrate the mini-project.
- Weightage of marks for PART A is 60% and for PART B is 40%. General rubrics suggested to be followed for part A and part B.
- Change of experiment is allowed only once and Marks allotted to the procedure part to be made zero (Not allowed for Part B).

The duration of SEE is 03 hours.

-

D1. ARTICULATION MATRIX

Mapping of CO to PO												
	POs											
COs	1	2	3	4	5	6	7	8	9	10	11	12
1. Create, Update and query on the database.	2	1	1	-	1	-	-	-	-	-	-	2
2. Demonstrate the working of different concepts of DBMS.	2	2	1	-	1	-	-	-	-	-	-	2
3. Implement, analyze and evaluate the project developed for an application.	1	3	3	-	3	-	-	-	2	-	2	2

Note: Mappings in the Tables D1 (above) and D2 (below) are done by entering in the corresponding cell the Correllation Levels in terms of numbers. For Slight (Low): 1, Moderate (Medium): 2, Substantial (High): 3 and for no correllation: " - ".

D2. ARTICULATION MATRIX CO v/s PSO

Mapping of CO to PSO			
	PSOs		
COs	1	2	3
1. Create, Update and query on the database.	-	3	-
2. Demonstrate the working of different concepts of DBMS.	-	3	-
3. Implement, analyze and evaluate the project developed for an application.	-	3	-

-

E. EXPERIMENTS

1. EXPERIMENT NO:1

2. TITLE: **LIBRARY DATABASE**

3. LEARNING OBJECTIVES:

- Become proficient in using database query language, i.e. SQL.
- Understand the issues related to database performance.

4. AIM:

Consider the following schema for Library Database:

BOOK(Book_id, Title, Publisher_Name, Pub_Year)

BOOK_AUTHORS(Book_id, Author_Name)

PUBLISHER(Name, Address, Phone)

BOOK_COPIES(Book_id, Programme_id, No-of_Copies)

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

LIBRARY_PROGRAMME(Programme_id, Programme_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, but from jan 2017 to jun 2017.
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.

https://www.youtube.com/watch?v=AaSU-AOguls

https://www.youtube.com/watch?v=-EwEvJxS-Fw

5. THEORY / HYPOTHESIS:

The sql create table statement

The create table statement is used to create a new table in a database.

syntax

```
create table table_name
(
        column1 datatype,
        column2 datatype,
        column3 datatype,
        ....
);
```

The column parameters specify the names of the columns of the table. The datatype parameter specifies the type of data the column can hold (e.g. varchar, integer, date, etc.).

Example

```
create table persons
(
        personid int,
        lastname varchar(255),
        firstname varchar(255),
        address varchar(255),
        city varchar(255)
        );
```

The personid column is of type int and will hold an integer. The lastname, firstname, address, and city columns are of type varchar and will hold characters, and the maximum length for these fields is 255 characters.

The sql drop table statement

The drop table statement is used to drop an existing table in a database.

syntax

```
        drop table table_name;
```

sql alter table statement

The alter table statement is used to add, delete, or modify columns in an existing table. the alter table statement is also used to add and drop various constraints on an existing table.

to add a column in a table, use the following syntax:

```
        alter table table_name
        add column_name datatype;
```

to delete a column in a table, use the following syntax :

```
alter table table_name
        drop column column_name;
```

to change the data type of a column in a table, use the following syntax:

```
        alter table table_name
        modify column_name datatype;
```

sql create constraints

Constraints can be specified when the table is created with the create table statement, or after the table is created with the alter table statement.

```
create table table_name
(
        column1 datatype constraint,
        column2 datatype constraint,
        column3 datatype constraint,
        ....
);
```

sql constraints are used to specify rules for the data in a table. The following constraints are commonly used in sql:

not null - ensures that a column cannot have a null value

unique - ensures that all values in a column are different

primary key - a combination of a not null and unique. uniquely identifies each row in a table

foreign key - uniquely identifies a row/record in another table

sql primary key constraint

The primary key constraint uniquely identifies each record in a database table. Primary keys must contain unique values, and cannot contain null values. A table can have only one primary key, which may consist of single or multiple fields.

```
	create table persons
	(
		id int,
		lastname varchar(255),
		firstname varchar(255),
		age int,
		primary key (id)
	);
```

sql primary key on alter table

To create a primary key constraint on the "id" column when the table is already created, use the following sql:

```
	alter table persons
	add primary key (id);
```

drop a primary key constraint

```
	alter table persons
	drop primary key;
```

6. PROCEDURE / PROGRAMME / ACTIVITY:

```
create database database_user_LIBRARY;
use database_user_LIBRARY;

create table publisher
(
	name varchar(10),
	address varchar(10),
	phone bigint,
	primary key(name)
);

insert into publisher values('mcgraw','noida','9085467001');

create table book
(
	book_id varchar(5),
	title varchar(20),
	publisher_name varchar(10),
	publisher_year int,
	primary key(book_id),
	foreign key(publisher_name) references publisher(name) on delete cascade
);

insert into book values('111','Management','mcgraw','2010');

create table book_authors
(
	book_id varchar(5),
	author_name varchar(15),
        primary key(book_id),
        foreign key(book_id) references book(book_id) on delete cascade
);

insert into book_authors values('111','tripathy reddy');


create table library_programme
(
        programme_id
        varchar(5),
        programme _name
        varchar(10),
        address varchar(15),
        primary key(programme_id)
);

insert into library_ programme values('11','vccampus','puttur');



create table book_copies
(
        book_id varchar(5),
        programme_id varchar(5),
        no_of_copies int,
        primary key(book_id, programme_id),
        foreign key(book_id)references book(book_id) on delete cascade,
        foreign key(programme_id) references library_ programme(programme_id) on
        delete cascade

);

insert into book_copies values('111','11',5);


create table book_lending
(
        book_id varchar(5),
        programme_id
        varchar(5),
        card_no varchar(5),
        date_out date,
        due_date date,
        primary key(book_id, programme_id,card_no),
        foreign key(book_id) references book(book_id)
        on delete cascade,
        foreign key(programme_id) references library_ programme(programme_id) on delete
        cascade
```

```
);


insert into book_lending values('111','11','1111','17-06-10','17-07-20');
```

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

```
select b.book_id, b.title, b.publisher_name, ba.author_name, bc. programme_id, bc.no_of_copies
from book b, book_authors ba, book_copies bc
where b.book_id=bc.book_id and b.book_id=ba.book_id;
```

2. Get the particulars of borrowers who have borrowed more than 3 books, but from Jan 2017 to Jun 2017.

```
        select distinct card_no
        from book_lending b
        where (date_out between '17-01-01' and '17-06-30')
        group by card_no having count(*)>3;
```

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

```
        delete from book where book_id='112';
```

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

```
create table book1
 (          book_id varchar(5),
            title varchar(20),
            publisher_name varchar(10),
            publisher_year int,
            primary key(book_id, publisher_year)
)


            partition by range (publisher_year)
                    (partition p1 values less than (2002),
                     partition p2 values less than (2010),
                     partition p3 values less than (maxvalue));
```

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

```
        create view available as
        (
        select book_id, sum(no_of_copies) -(select count(card_no)
                                            from book_lending
                                            where b.book_id = book_id) as avail_copies
        from book_copies b
        group by book_id
        );
```

7. RESULTS & CONCLUSIONS:

```
select * from publisher;
```

NAME	ADDRESS	PHONE
mcgraw	noida	9085467001
phi	pune	9945467800
pearson	nagpur	7875622333

```
select * from book;
```

BOOK_ID	TITLE	PUBLISHER_NAME	PUBLISHER_YEAR
111	management	mcgraw	2010
112	computer networks	pearson	2006
113	database concepts	pearson	2014
115	entrepreneurship	pearson	2010
114	formal languages	mcgraw	2006
116	embedded systems	mcgraw	2014
117	programming in java	phi	2010

```
select * from book_authors;
```

BOOK_ID	AUTHOR_NAME
111	tripathy reddy
112	larry peterson
113	Ramez navathe
114	john e hopcroft
115	vasant desai
116	rajkamal
117	herbert schildt

```
select * from library_branch;
```

BRANCH_ID	BRANCH_NAME	ADDRESS
11	vc campus	puttur
12	pvs	mangalore
13	mg road	bangalore

```
select * from book_copies;
```

BOOK_ID	BRANCH_ID	NO_OF_COPIES
111	11	5

111	13	10
112	12	5
113	11	12
113	12	20
114	13	15
115	11	9
115	13	25
116	13	5
117	12	5

select * from book_lending;

BOOK_ID	BRANCH_ID	CARD_NO	DATE_OUT	DUE_DATE
111	11	1111	17-06-10	17-07-20
111	13	1112	17-07-13	17-07-23
114	13	1113	17-06-05	17-07-15
115	13	1113	17-06-10	17-06-20
116	13	1113	17-06-15	17-07-25
111	13	1113	17-03-23	17-04-02
111	13	1114	17-03-20	17-03-30
113	11	1111	17-04-02	17-04-12
113	12	1111	17-05-05	17-05-05
115	11	1111	17-02-02	17-02-12

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

BOOK_ID	TITLE	PUBLISHER_NAME	AUTHOR_NAME	BRANCH_ID	NO_OF_COPIES
111	management	mcgraw	tripathy reddy	11	5
111	management	mcgraw	tripathy reddy	13	10
112	computer networks	pearson	larry peterson	12	5
113	database concepts	pearson	ramez navathe	11	12
113	database concepts	pearson	ramez navathe	12	20
114	formal languages	mcgraw	john e hopcroft	13	15
115	entrepreneurship	pearson	vasant desai	11	9
115	entrepreneurship	pearson	vasant desai	13	25
116	embedded systems	mcgraw	rajkamal	13	5

117	programming in java	phi	herbert schildt	12	5

2. Get the particulars of borrowers who have borrowed more than 3 books, but from Jan 2017 to Jun 2017.

CARD_NO
1113
1111

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

BOOK_ID	TITLE	PUBLISHER_NAME	PUBLISHER_YEAR
111	management	mcgraw	2010
113	database concepts	pearson	2014
115	entrepreneurship	pearson	2010
114	formal languages	mcgraw	2006
116	embedded systems	mcgraw	2014
117	programming in java	phi	2010

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

```
select * from book1 partition(p1);
select * from book1 partition(p2);
select * from book1 partition(p3);
```

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

```
select * from available;
```

BOOK_ID	AVAIL_COPIES
113	30
115	32
117	5
112	5
116	4
111	11
114	14

8. LEARNING OUTCOMES :

- Understand basic concepts of dbms such as create table, select statement, where clause, views, aliasing.
- Understand entity integrity constraint and referential integrity constraint.
- Understand the usage of different data types.

9. APPLICATION AREAS:

- Design and develop database applications for real world problems such as health care, education, industry, transport, supply chain etc.

10. REMARKS:

1. EXPERIMENT NO:2

2. TITLE: **ORDER DATABASE**

3. LEARNING OBJECTIVES:

- Be familiar with a broad range of database management issues including data integrity, security, and recovery.
- Be able to apply proper techniques, such as normalization, in designing a database.

4. AIM:

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 Bangalore's average.
2. Find the name and numbers of all salesman who had more than one customer.
3. List all the salesman 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.

5. THEORY / HYPOTHESIS:

sql foreign key constraint

A foreign key is a key used to link two tables together. A foreign key is a field (or collection of fields) in one table that refers to the primary key in another table. The table containing the foreign key is called the child table, and the table containing the candidate key is called the referenced or parent table.

look at the following two tables:

"persons" table:

personid	lastname	Firstname	age
1	hansen	ola	30
2	svendson	tove	23
3	pettersen	kari	20

orders" table:

orderid	ordernumber	personid
1	77895	3
2	44678	3
3	22456	2

4	24562	1

Notice that the "personid" column in the "orders" table points to the "personid" column in the "persons" table. The "personid" column in the "persons" table is the primary key in the "persons" table. The "personid" column in the "orders" table is a foreign key in the "orders" table.

The foreign key constraint is used to prevent actions that would destroy links between tables. The foreign key constraint also prevents invalid data from being inserted into the foreign key column, because it has to be one of the values contained in the table it points to.

sql foreign key on create table

The following sql creates a foreign key on the "personid" column when the "orders" table is created:

```
create table orders
(
        orderid int not null,
        ordernumber int not null,
        personid int,
        primary key (orderid),
        foreign key (personid) references persons(personid)
);
```

sql foreign key on alter table

To create a foreign key constraint on the "personid" column when the "orders" table is already created, use the following sql:

```
alter table orders
add foreign key (personid) references persons(personid);
```

The sql insert into statement

The insert into statement is used to insert new records in a table. It is possible to write the insert into statement in two ways.

The first way specifies both the column names and the values to be inserted:

```
insert into table_name (column1, column2, column3, ...) values (value1, value2, value3, ...);
```

If you are adding values for all the columns of the table, you do not need to specify the column names in the sql query. However, make sure the order of the values is in the same order as the columns in the table. The insert into syntax would be as follows:

```
insert into table_name values (value1, value2, value3, ...);
```

The sql select statement

The select statement is used to select data from a database. The data returned is stored in a result table, called the result-set.

```
select column1, column2, ...
from table_name;
```

Here, column1, column2, ... are the field names of the table you want to select data from. If you want to select all the fields available in the table, use the following syntax:

```
select * from table_name;
```

The sql where clause

The where clause is used to filter records. The where clause is used to extract only those records that fulfill a specified condition.

```
        select column1, column2, ...
        from table_name
        where condition;
```

6. PROCEDURE / PROGRAMME / ACTIVITY:

```
create database database_user_ORDER;
use database_user_ORDER;

create table salesman
(
        salesman_id varchar(5),
        name varchar(15),
        city varchar(15),
        commission int,
        primary key(salesman_id)
);

insert into salesman values('1','Guru','Mangalore',5);


create table customer
(
        customer_id varchar(5),
        cust_name varchar(15),
        city varchar(15),
        grade int,
        salesman_id varchar(5),
        primary key(customer_id),
        foreign key(salesman_id) references salesman(salesman_id) on delete cascade
);

insert into customer values('C11','Srikanth','Bangalore',4,'2');


create table orders
(
        ord_no varchar(5),
        purchase_amt int,
        ord_date date,
        customer_id varchar(5),
        salesman_id varchar(5),
        primary key(ord_no),
        foreign key(customer_id) references customer(customer_id) on delete cascade,
        foreign key(salesman_id) references salesman(salesman_id) on delete cascade
);

insert into orders values('O111',2500,'17-07-11','C11','2');
```

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

```
select count(*) as count
from customer where grade >
        (select avg(grade)
        from customer
        where city ='Bangalore');
```

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

```
select s.salesman_id, s.name, count(customer_id)
from salesman s, customer c
where s.salesman_id = c.salesman_id
group by s.salesman_id, s.name
having count(customer_id)>1;
```

3. List all the salesman and indicate those who have and don't have customers in their cities (Use UNION operation.)

```
select name,'exists' as same_city
from salesman s
where city in
        (select city
        from customer
        where s.salesman_id = salesman_id)
union
select name,'not exists' as same_city
from salesman s where
city not in
        (select city
        from customer
        where s.salesman_id = salesman_id);
```

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

```
create view highest_order as
select s.salesman_id, s.name,o.purchase_amt,o.ord_date
from salesman s,orders o
where s.salesman_id = o.salesman_id;

select name,ord_date
from highest_order h
where purchase_amt =
        (select max(purchase_amt)
        from highest_order
        where h.ord_date = ord_date);
```

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

```
delete from salesman where salesman_id =3;
```

7. RESULTS & CONCLUSIONS:

select * from salesman;

SALESMAN_ID	NAME	CITY	COMMISSION
1	Guru	Mangalore	5
2	Ravi	Bangalore	3
3	Girish	Hubli	3
4	Sagar	Bangalore	3
5	Raj	Mangalore	4

select * from customer;

CUSTOMER_ID	CUST_NAME	CITY	GRADE	SALESMAN_ID
C11	Srikanth	Bangalore	4	2
C12	Sandeep	Mangalore	2	3
C13	Uday	Bangalore	3	2
C14	Mahesh	Hubli	2	2
C15	Shivaram	Bangalore	2	3
C16	Shyam	Mangalore	5	1
C17	Sumith	Udupi	4	5
C18	Shravan	Bangalore	3	4

select * from orders;

ORD_NO	PURCHASE_AMT	ORD_DATE	CUSTOMER_ID	SALESMAN_ID
O111	2500	17-07-11	C11	2
O112	1999	17-07-09	C12	3
O113	999	17-07-12	C13	2
O114	9999	17-07-12	C14	2
O115	7999	17-07-11	C15	3
O116	1099	17-07-09	C16	1

1. Count the customers with grades above Bangalore’s average.

COUNT
3

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

SALESMAN_ID	NAME	COUNT(CUSTOMER_ID)
2	Ravi	3
3	Girish	2

3. List all the salesman and indicate those who have and don’t have customers in their cities (Use UNION operation.)

NAME	SAME_CITY
Girish	not exists
Guru	exists
Raj	Not exists
Ravi	Exists
Sagar	Exists

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

NAME	ORD_DATE
Girish	17-07-09
Girish	17-07-11
Ravi	17-07-12

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

SALESMAN_ID	NAME	CITY	COMMISSION
1	Guru	Mangalore	5
2	Ravi	Bangalore	3
4	Sagar	Bangalore	3
5	Raj	Mangalore	4

CUSTOMER_ID	CUST_NAME	CITY	GRADE	SALESMAN_ID
C11	Srikanth	Bangalore	4	2
C13	Uday	Bangalore	3	2
C14	Mahesh	Hubli	2	2
C16	Shyam	Mangalore	5	1
C17	Sumith	Udupi	4	5
C18	Shravan	Bangalore	3	4

ORD_NO	PURCHASE_AMT	ORD_DATE	CUSTOMER_ID	SALESMAN_ID
O111	2500	17-07-11	C11	2
O113	999	17-07-12	C13	2
O114	9999	17-07-12	C14	2
O116	1099	17-07-09	C16	1

8. LEARNING OUTCOMES :

- Understand aggregate functions such as count, avg, max.
- Understand the usage of group by and having.
- Understand the usage of union, in and not in.

9. APPLICATION AREAS:

- Design and develop database applications for real world problems such as health care, education, industry, transport, supply chain etc.

10. REMARKS:

-

1. EXPERIMENT NO:3

2. TITLE: **MOVIE DATABASE**

3. LEARNING OBJECTIVES:

- Be able to use several commercially available database management systems such as Access and Oracle SQL Plus.
- Be able to use advanced SQL to create, manipulate, and query databases.

4. AIM:

Consider the schema for Movie Database:
ACTOR(Act_id, Act_Name, Act_Gender)
DIRECTOR(Dir_id, Dir_Name, Dir_Phone)
MOVIES(Mov_id, Mov_Title, Mov_Year, Mov_Lang, Dir_id)
MOVIE_CAST(Act_id, Mov_id, Role)
RATING(Mov_id, Rev_Stars)

Write SQL queries to
1. List the titles of all movies directed by 'Hitchcock'.
2. Find the movie names where one or more actors acted in two or more movies.
3. List all actors who acted in a movie before 2000 and also in a movie after 2015 (use JOIN operation).
4. Find the title of movies and number of stars for each movie that has at least one rating and find the highest number of stars that movie received. Sort the result by movie title.
5. Update rating of all movies directed by 'Steven Spielberg' to 5.

5. THEORY / HYPOTHESIS:

The sql and, or and not operators

The where clause can be combined with and, or, and not operators. The and and or operators are used to filter records based on more than one condition:

The and operator displays a record if all the conditions separated by and is true.
The or operator displays a record if any of the conditions separated by or is true.
The not operator displays a record if the condition(s) is not true.

and syntax

```
select column1, column2, ...
from table_name
where condition1 and condition2 and condition3 ...;
```

or syntax

```
        select column1, column2, ...
        from table_name
        where condition1 or condition2 or condition3 ...;
```

not syntax

```
        select column1, column2, ...
        from table_name
        where not condition;
```

You can also combine the and, or and not operators.

The sql order by keyword

The order by keyword is used to sort the result-set in ascending or descending order. The order by keyword sorts the records in ascending order by default. To sort the records in descending order, use the desc keyword.

```
        select column1, column2, ...
        from table_name
        order by column1, column2, ... asc|desc;
```

sql null values

what is a null value?

A field with a null value is a field with no value. If a field in a table is optional, it is possible to insert a new record or update a record without adding a value to this field. Then, the field will be saved with a null value. It is not possible to test for null values with comparison operators, such as =, <, or <>. We will have to use the is null and is not null operators instead.

is null syntax

```
        select column_names
        from table_name
        where column_name is null;
```

is not null syntax

```
        select column_names
        from table_name
        where column_name is not null;
```

6. PROCEDURE / PROGRAMME / ACTIVITY:

```
create database database_user_MOVIE;
use database_user_MOVIE;

create table actor
(
        act_id varchar(5),
        act_name varchar(15),
        act_gender varchar(6),
        primary key(act_id)
);

insert into actor values('A101','Raj','M');
```

```
create table director
(
        dir_id varchar(5),
        dir_name varchar(15),
        dir_phone bigint,
        primary key(dir_id)
);

insert into director values('D01','Hitchcock',8723268423);

create table movies
(
        mov_id varchar(5),
        mov_title varchar(20),
        mov_year int,
        mov_lang varchar(10),
        dir_id varchar(5),
        primary key(mov_id),
        foreign key(dir_id)  references director(dir_id) on delete cascade
);

insert into movies values('M10','Psycho',1960,'english','D01');

create table movie_cast
(
        act_id varchar(5),
        mov_id varchar(5),
        role varchar(10),
        primary key(act_id, mov_id),
        foreign key(act_id) references actor(act_id) on delete cascade,
        foreign key(mov_id) references movies(mov_id) on delete cascade
);

insert into movie_cast values('A101','M11','m_lead');

create table rating
(
        rat_id varchar(5),
        mov_id varchar(5),
        rev_stars int,
        primary key(rat_id),
        foreign key(mov_id) references movies(mov_id) on delete cascade
);

insert into rating values('R1','M11',4);
```

1. List the titles of all movies directed by 'Hitchcock'.

```
        select mov_title
```

```
          from movies m, director d
          where m.dir_id = d.dir_id and d.dir_name ='Hitchcock';
```

2. Find the movie names where one or more actors acted in two or more movies.

```
          select distinct mov_title from movies m, movie_cast mc
          where m.mov_id = mc.mov_id and
                    (select count(mov_id)
                    from movie_cast
                    where act_id =mc.act_id)>=2;
```

3. List all actors who acted in a movie before 2000 and also in a movie after 2015 (use JOIN operation).

```
          select act_name
          from actor a join movie_cast mc on a.act_id = mc.act_id join movies m
          on mc.mov_id = m.mov_id
          where m.mov_year<2000
          and act_name in (
                              select act_name
                              from actor a join movie_cast mc on a.act_id = mc.act_id join movies m
                              on mc.mov_id = m.mov_id
                              where m.mov_year>2015);
```

4. Find the title of movies and number of stars for each movie that has at least one rating and find the highest number of stars that movie received. Sort the result by movie title.

```
          select mov_title, max(rev_stars)
          from movies m, rating r
          where m.mov_id = r.mov_id group by m.mov_title order by m.mov_title;
```

5. Update rating of all movies directed by 'Steven Spielberg' to 5

```
          update rating set rev_stars=5
          where mov_id in
                    (select m.mov_id
                    from movies m, director d
                    where m.dir_id = d.dir_id and d.dir_name='Steven Spielberg');
          select  * from rating;
```

7. RESULTS & CONCLUSIONS:

select * from actor;

ACT_ID	ACT_NAME	ACT_GENDER
A101	Raj	M
A102	Johny	M
A103	Leo	M
A104	Saru	F
A105	Jasmine	F
A106	Anthony parkins	M

A107	Harrison ford	M

select * from director;

DIR_ID	DIR_NAME	DIR_PHONE
D01	Hitchcock	8723268423
D02	Steven spielberg	8938732432
D03	Rajamouli	9434784454
D04	Nraj	9342400533
D05	Pawan	8757563322

select * from movies;

MOV_ID	MOV_TITLE	MOV_YEAR	MOV_LANG	DIR_ID
M10	Psycho	1960	english	D01
M11	Tomorrow comes now	2017	english	D04
M12	Its a crime	1999	english	D04
M13	Indiana jones and the temples of doom	1984	english	D02
M14	Hello hello	2016	english	D04
M15	E.T.	1982	english	D02

select * from movie_cast;

ACT_ID	MOV_ID	ROLE
A101	M11	m_lead
A104	M11	f_lead
A101	M12	m_lead
A106	M10	negative
A107	M13	m_lead
A104	M14	f_lead
A107	M14	supporting

select * from rating;

RAT_ID	MOV_ID	REV_STARS
R1	M11	4
R2	M10	4
R3	M11	3
R4	M12	4

R5	M13	4
R6	M15	3
R7	M13	3

1. List the titles of all movies directed by ‘Hitchcock’.

MOV_TITLE
Psycho

2. Find the movie names where one or more actors acted in two or more movies.

MOV_TITLE
Hello hello
Indiana jones and the temples of doom
Its a crime
Tomorrow comes now

3. List all actors who acted in a movie before 2000 and also in a movie after 2015 (use JOIN operation).

ACT_NAME
Harrison ford
Raj

4. Find the title of movies and number of stars for each movie that has at least one rating and find the highest number of stars that movie received. Sort the result by movie title.

MOV_TITLE	MAX(REV_STARS)
E.T.	3
Indiana jones and the temples of doom	4
Its a crime	4
Psycho	4
Tomorrowcomesnow	4

5. Update rating of all movies directed by ‘Steven Spielberg’ to 5.

RAT_ID	MOV_ID	REV_STARS
R1	M11	4
R2	M10	4
R3	M11	3
R4	M12	4

R5	M13	5
R6	M15	5
R7	M13	5

8. LEARNING OUTCOMES :

- Understand the usage of intersect, order by and update.

9. APPLICATION AREAS:

- Design and develop database applications for real world problems such as health care, education, industry, transport, supply chain etc.

10. REMARKS:

-

1. EXPERIMENT NO:4

2. TITLE: **COLLEGE DATABASE**

3. LEARNING OBJECTIVES:

- Foundation knowledge in database concepts, technology and practice to groom students into well-informed database application developers.

4. AIM:

Consider the schema for College Database:
STUDENT(USN, SName, Address, Phone, Gender)
SEMSEC(SSID, Sem, Sec)
CLASS(USN, SSID)
COURSE(Subcode, Title, Sem, Credits)
IAMARKS(USN, Subcode, SSID, Test1, Test2, Test3, FinalIA)

Write SQL queries to
1. List all the student details studying in fourth semester ‘C’ section.
2. Compute the total number of male and female students in each semester and in each section.
3. Create a view of Test1 marks of student USN ‘1BI15CS101’ in all Courses.
4. Calculate the FinalIA (average of best two test marks) and update the corresponding table for all students.
5. Categorize students based on the following criterion:
If FinalIA = 17 to 20 then CAT = ‘Outstanding’
If FinalIA = 12 to 16 then CAT = ‘Average’
If FinalIA< 12 then CAT = ‘Weak’
Give these details only for 8 th semester A, B, and C section students.

5. THEORY / HYPOTHESIS:

The sql update statement

The update statement is used to modify the existing records in a table.

update syntax

```
        update table_name
        set column1 = value1, column2 = value2, ...
        where condition;
```

Notice the where clause in the update statement. The where clause specifies which record(s) that should be updated. if you omit the where clause, all records in the table will be updated.

The sql delete statement

The delete statement is used to delete existing records in a table.

delete syntax

```
        delete from table_name
        where condition;
```

The where clause specifies which record(s) that should be deleted. If you omit the where clause, all records in the table will be deleted.

The sql min() and max() functions

The min() function returns the smallest value of the selected column. The max() function returns the largest value of the selected column.

min() syntax

```
        select min(column_name)
        from table_name
        where condition;
```

max() syntax

```
        select max(column_name)
        from table_name
        where condition;
```

sql count(), avg() and sum() functions

The count() function returns the number of rows that matches a specified criteria. The avg() function returns the average value of a numeric column. The sum() function returns the total sum of a numeric column.

count() syntax

```
        select count(column_name) from table_name where condition;
```

avg() syntax

```
        select avg(column_name) from table_name where condition;
```

sum() syntax

```
        select sum(column_name) from table_name  where condition;
```

The sql like operator

The like operator is used in a where clause to search for a specified pattern in a column. There are two wildcards used in conjunction with the like operator:

% - the percent sign represents zero, one, or multiple characters
_ - the underscore represents a single character

The percent sign and the underscore can also be used in combinations.

```
        select column1, column2, ...
        from table_name
        where columnn like pattern;
```

The sql in operator

The in operator allows you to specify multiple values in a where clause. The in operator is a shorthand for multiple or conditions.

```
        select column_name(s)
        from table_name
        where column_name in (value1, value2, ...);
                        or
        select column_name(s)
        from table_name
        where column_name in (select statement);
```

sql between operator

The between operator selects values within a given range. The values can be numbers, text, or dates. The between operator is inclusive: begin and end values are included.

```
        select column_name(s)
        from table_name
        where column_name between value1 and value2;
```

6. PROCEDURE / PROGRAMME / ACTIVITY:

```
create database database_user_COLLEGE;
use database_user_COLLEGE;

create table student
(
        usn varchar(10),
        sname varchar(15),
        address varchar(15),
        phone bigint,
        gender varchar(6),
        primary key(usn)
);

insert into student values('1BI15CS100','Namitha','Udupi',7860054110,'f');

create table semsec
(
        ssid varchar(5),
        sem int,
        sec varchar(1),
        primary key(ssid)
);

insert into semsec values('S01',1,'A');

create table class
(
        usn varchar(10),
        ssid varchar(5),
        primary key(usn),
        foreign key(usn) references student(usn) on delete cascade,
        foreign key(ssid) references semsec(ssid) on delete cascade
);

insert into class values('1BI15CS100','S04');

create table course
(
        subcode varchar(7),
        title varchar(15),
        sem int,
        credits int,
        primary key(subcode)
);

insert into subject values('15CS14','Algorithms',1,4);

create table iamarks
(
        usn varchar(10),
        subcode varchar(7),
        ssid varchar(5),
        test1 int,
        test2 int,
        test3 int,
        final_ia int,
        primary key(usn,subcode,ssid),
        foreign key(usn) references student(usn) on delete cascade,
        foreign key(subcode) references subject(subcode) on delete cascade,
        foreign key(ssid) references semsec(ssid)  on delete cascade
);

insert into iamarks values('1BI15CS100','15CS41','S05',19,18,20,NULL);
```

1. List all the student details studying in fourth semester ‘C’ section.

```
        select s.usn, sname, gender, address
        from student s, semsec sc ,class c
        where s.usn=c.usn and c.ssid= sc.ssid and sc.sem = 4 and sc.sec ='C';
```

2. Compute the total number of male and female students in each semester and in each section.

```
        select sem, sec, gender, count(*) as count
        from student s, semsec sc, class c
        where s.usn = c.usn and sc.ssid = c.ssid
        group by sem, sec, gender;
```

3. Create a view of Test1 marks of student USN ‘1BI15CS101’ in all subjects.

```
        create view test1_marks as
        (
                select usn, test1, subcode
                from iamarks
                where usn='1BI15CS101'
        );
```

4. Calculate the FinalIA (average of best two test marks) and update the corresponding table for all students.

```
        create table average_finder
        (
                select usn,subcode,greatest(test1,test2,test3) as highest, case
                when test1<greatest(test1,test2,test3) and test1>least(test1,test2,test3) then test1
                when test2<greatest(test1,test2,test3) and test2>least(test1,test2,test3) then test2
                else test3
                end as second_highest from iamarks
        );

        update iamarks a set final_ia =
                (select (highest+second_highest)/2 from average_finder
                where a.usn =usn and a.subcode= subcode);
```

5. Categorize students based on the following criterion:
If FinalIA = 17 to 20 then CAT = ‘Outstanding’
If FinalIA = 12 to 16 then CAT = ‘Average’
If FinalIA< 12 then CAT = ‘Weak’
Give these details only for 8 th semester A, B, and C section students.

```
        select usn, subcode, case
        when final_ia>=17 and final_ia<=20 then 'Outstanding'
        when final_ia>=12 and final_ia<=16 then 'Average'
        when final_ia<12 then 'Weak'
        end as category
        from iamarks
        where usn in
                (select usn from semsec sc, class c where sc.ssid=c.ssid and sem=8 and sec in
(‘A’,’B’,’C’));
```

7. RESULTS & CONCLUSIONS:
select * from student;

USN	SNAME	ADDRESS	PHONE	GENDER
1BI15CS100	Namitha	Udupi	7860054110	f
1BI15CS101	Mithun	Virajpet	8762514991	m
1BI15CS102	Kshama	Puttur	9000876123	f
1BI15CS103	Raghavendra	Karwar	8700967408	m
1BI15CS104	Sooraj	Bangalore	7773334422	m
1BI15CS105	Karthik	Puttur	7789086125	m

select * from semsec;

SSID	SEM	SEC
S01	1	A
S04	4	A
S05	4	C
S06	8	A
S07	8	B
S08	8	C

select * from class;

USN	SSID
1BI15CS100	S04
1BI15CS101	S05
1BI15CS102	S01
1BI15CS103	S06
1BI15CS104	S07
1BI15CS105	S08

select * from course;

SUBCODE	TITLE	SEM	CREDITS
15CS14	Algorithms	1	4
15CS41	Graph theory	4	3
15CS43	Processors	4	4
15CS81	Oop with c++	8	4
15CS82	Networks	8	4
15CS83	DBMS	8	3

select * from iamarks;

USN	SUBCODE	SSID	TEST1	TEST2	TEST3	FINALIA
1BI15CS100	15CS41	S05	19	18	20	
1BI15CS101	15CS43	S04	15	18	19	
1BI15CS101	15CS41	S04	15	17	14	
1BI15CS102	15CS14	S01	10	11	8	
1BI15CS103	15CS14	S01	13	17	15	
1BI15CS104	15CS81	S08	13	17	19	
1BI15CS104	15CS82	S06	12	9	10	
1BI15CS105	15CS81	S07	19	17	16	
1BI15CS105	15CS83	S08	19	17	18	

1. List all the student details studying in fourth semester 'C' section.

USN	SNAME	GENDER	ADDRESS
1BI15CS101	Mithun	m	Virajpet

2. Compute the total number of male and female students in each semester and in each section.

SEM	SEC	GENDER	COUNT
1	A	f	1
4	A	f	1
4	C	m	1
8	A	m	1
8	B	m	1
8	C	m	1

3. Create a view of Test1 marks of student USN '1BI15CS101' in all subjects.

select * from test1_marks;

USN	TEST1	SUBCODE
1BI15CS101	15	15CS43
1BI15CS101	15	15CS41

4. Calculate the FinalIA (average of best two test marks) and update the corresponding table for all students.

USN	SUBCODE	SSID	TEST1	TEST2	TEST3	FINAL_IA
1BI15CS100	15CS41	S05	19	18	20	20
1BI15CS101	15CS43	S04	15	18	19	19
1BI15CS101	15CS41	S04	15	17	14	16
1BI15CS102	15CS14	S01	10	11	8	11
1BI15CS103	15CS14	S01	13	17	15	16
1BI15CS104	15CS81	S08	13	17	19	18
1BI15CS104	15CS82	S06	12	9	10	11
1BI15CS105	15CS81	S07	19	17	16	18
1BI15CS105	15CS83	S08	19	17	18	19

5. Categorize students based on the following criterion:
If FinalIA = 17 to 20 then CAT = 'Outstanding'
If FinalIA = 12 to 16 then CAT = 'Average'
If FinalIA< 12 then CAT = 'Weak'
Give these details only for 8 th semester A, B, and C section students.

USN	SUBCODE	CATEGORY

1BI15CS103	15CS14	Average
1BI15CS104	15CS81	Outstanding
1BI15CS104	15CS82	Weak
1BI15CS105	15CS81	Outstanding
1BI15CS105	15CS83	Outstanding

8. LEARNING OUTCOMES :

- Understand the usage of case for categorizing data in the table.

9. APPLICATION AREAS:

- Design and develop database applications for real world problems such as health care, education, industry, transport, supply chain etc.

10. REMARKS:

-

1. EXPERIMENT NO:5

2. TITLE: **COMPANY DATABASE**

3. LEARNING OBJECTIVES:

- Strong practice in SQL programming through a variety of database problems.

4. AIM:

Consider the schema for Company Database:
EMPLOYEE(SSN, Name, Address, Sex, Salary, SuperSSN, DNo)
DEPARTMENT(DNo, DName, MgrSSN, MgrStartDate)
DLOCATION(DNo, DLoc)
PROJECT(PNo, PName, PLocation, DNo)
WORKS_ON(SSN, PNo, Hours)

Write SQL queries to
1. Make a list of all project numbers for projects that involve an employee whose last name is ‘Scott’, either as a worker or as a manager of the department that controls the project.
2. Show the resulting salaries if every employee working on the ‘IoT’ project is given a 10 percent raise.
3. Find the sum of the salaries of all employees of the ‘Accounts’ department, as well as the maximum salary, the minimum salary, and the average salary in this department
4. Retrieve the name of each employee who works on all the projects controlled by department number 5 (use NOT EXISTS operator).
5. For each department that has more than five employees, retrieve the department number and the number of its employees who are making more than Rs. 6,00,000.

5. THEORY / HYPOTHESIS:

sql aliases

sql aliases are used to give a table, or a column in a table, a temporary name. Aliases are often used to make column names more readable. An alias only exists for the duration of the query.

alias column syntax

```
select column_name as alias_name
from table_name;
```

alias table syntax

```
select column_name(s)
from table_name as alias_name;
```

Different types of sql joins

Here are the different types of the joins in sql:

(inner) join: returns records that have matching values in both tables

left (outer) join: return all records from the left table, and the matched records from the right table

right (outer) join: return all records from the right table, and the matched records from the left table

full (outer) join: return all records when there is a match in either left or right table

The sql union operator

The union operator is used to combine the result-set of two or more select statements.

- each select statement within union must have the same number of columns
- the columns must also have similar data types
- the columns in each select statement must also be in the same order

```
select column_name(s) from table1
union
select column_name(s) from table2;
```

The sql group by statement

The group by statement is often used with aggregate functions (count, max, min, sum, avg) to group the result-set by one or more columns.

```
select column_name(s)
from table_name
where condition
group by column_name(s);
```

The sql having clause

The having clause was added to sql because the where keyword could not be used with aggregate functions.

```
select column_name(s)
from table_name
where condition
group by column_name(s)
having condition;
```

The sql exists operator

The exists operator is used to test for the existence of any record in a subquery. The exists operator returns true if the subquery returns one or more records.

```
        select column_name(s)
        from table_name
        where exists
        (select column_name from table_name where condition);
```

The sql any and all operators

The any and all operators are used with a where or having clause. The any operator returns true if any of the subquery values meet the condition. The all operator returns true if all of the subquery values meet the condition.

any syntax

```
        select column_name(s)
        from table_name
        where column_name operator any
        (select column_name from table_name where condition);
```

all syntax

```
        select column_name(s)
        from table_name
        where column_name operator all
        (select column_name from table_name where condition);
```

note: the operator must be a standard comparison operator (=, <>, !=, >, >=, <, or <=).

6. PROCEDURE / PROGRAMME / ACTIVITY:

```
create table department
(
        dno varchar(5),
        dname varchar(15),
        mgrssn varchar(5),
        mgrstartdate date,
        primary key(dno),
);

create table employee
(
        ssn varchar(5),
        name varchar(15),
        address varchar(15),
        sex varchar(6),
        salary int,
        superssn varchar(5),
        dno varchar(5),
        primary key(ssn),
);

alter table employee add constraint fk1 foreign key(dno) references department(dno) on delete cascade;

alter table employee add constraint fk2 foreign key(superssn) references employee(ssn) on delete cascade
alter table department add constraint fk3 foreign key(mgrssn) references employee(ssn) on delete cascade

insert into department values('&dno','&dname','&mgrssn','&mgrstartdate');

insert into employee values('&ssn','&name','&address','&sex','&salary','&superssn','&dno');

create table dlocation
(
        dno varchar(5),
        dloc varchar(15),
        primary key (dno,dloc),
        foreign key(dno) references department(dno) on delete cascade
);

insert into dlocation values('&dno','&dloc');

create table project
(
        pno varchar(5),
        pname varchar(10),
        plocation varchar(10),
        dno varchar(5),
        primary key(pno),
        foreign key(dno) references department(dno) on delete cascade
);

insert into project values('&pno','&pname','&plocation','&dno');

create table works_on
(
        ssn varchar(5),
        pno varchar(5),
        hours int,
        primary key(ssn,pno),
        foreign key(ssn) references employee(ssn) on delete cascade,
        foreign key(pno) references project(pno) on delete cascade
);

insert into works_on values('&ssn','&pno','&hours');
```

1. Make a list of all project numbers for projects that involve an employee whose last name is ‘Scott’, either as a worker or as a manager of the department that controls the project.

```
        select distinct pno
        from project
        where pno in
                (select  pno
                from project p, department d, employee e
                where p.dno = d.dno and d.mgrssn = e.ssn and name like '%Scott')
              or  pno in
                (select pno
                from works_on w, employee e
                where w.ssn = e.ssn and name like '%Scott');
```

2. Show the resulting salaries if every employee working on the 'IoT' project is given a 10 percent raise.

```
    select e.name, e.salary*1.1 as new_salary
    from  employee e, works_on w
    where e.ssn = w.ssn and
    w.pno in
        (select pno
        from project
        where pname ='IoT');
```

3. Find the sum of the salaries of all employees of the 'Accounts' department, as well as the maximum salary, the minimum salary, and the average salary in this department.

```
    select sum(salary), max(salary), min(salary), avg(salary)
    from (employee e join department d on d.dno = e.dno)
    where d.dname = 'Accounts';
```

4. Retrieve the name of each employee who works on all the projects controlledby department number 5 (use NOT EXISTS operator).

```
    select name from employee e
    where not exists ((select pno from project where dno=5) minus
    (select pno from works_on where ssn = e.ssn));
```

5. For each department that has more than five employees, retrieve the department number and the number of its employees who are making more than Rs.6,00,000.

```
    select d.dno,count(*) as count
    from department d,employee e
    where d.dno= e.dno and salary >600000
    and d.dno in
        (select dno from employee
        group by dno having count(*)>5)
        group by d.dno;
```

7. RESULTS & CONCLUSIONS:

select * from department;

DNO	DNAME	MGRSSN	MGRSTARTDATE
1	Account	E107	10-jan-15
2	Research	E103	13-jul-16
3	Administration	E102	20-jul-17
4	Headquarters	E104	17-aug-16
5	Marketing	E106	28-apr-16

select * from employee;

SSN	NAME	ADDRESS	SEX	SALARY	SUPERSSN	DNO
E101	Allan Scott	Bengaluru	m	500000	E103	3
E102	Kishore	Bengaluru	m	620000	-	1
E103	Jimmy Scott	Mumbai	m	630000	-	2
E104	Leo	Delhi	m	650000	E107	3
E105	Joseph	Bengaluru	m	500000	E102	3
E106	Somashekhar	Chennai	m	550000	E107	3
E107	Rajkumar	Bengaluru	m	700000	-	1
E108	Kajal	Delhi	f	650000	-	3
E109	Smrithi	Bengaluru	f	620000	E108	3

select * from dlocation;

DNO	DLOC
1	Bengaluru
2	Bengaluru
3	Chennai
4	Hyderabad
5	Bengaluru
3	Bengaluru

select * from project;

PNO	PNAME	PLOCATION	DNO
P1	IoT	Bengaluru	5
P2	Android	Bengaluru	5
P3	Web	Chennai	3
P4	HTML	Hyderabad	2
P5	Medical	Hyderabad	2

select * from works_on;

SSN	PNO	HOURS
E101	P1	5
E104	P1	6
E108	P1	7
E105	P3	6
E105	P1	4
E104	P2	5

1. Make a list of all project numbers for projects that involve an employee whose last name is 'Scott', either as a worker or as a manager of the department that controls the project.

PNO
P1
P4
P5

2. Show the resulting salaries if every employee working on the 'IoT' project is given a 10 percent raise.

NAME	NEW_SALARY
Kajal	715000
Leo	715000
Joseph	550000
Allan Scott	550000

3. Find the sum of the salaries of all employees of the 'Accounts' department, as well as the maximum salary, the minimum salary, and the average salary in this department.

SUM(SALARY)	MAX(SALARY)	MIN(SALARY)	AVG(SALARY)
1320000	700000	620000	660000

4. Retrieve the name of each employee who works on all the projects controlled by department number 5 (use NOT EXISTS operator).

NAME
Leo

5. For each department that has more than five employees, retrieve the department number and the number of its employees who are making more than Rs. 6,00,000.

DNO	COUNT
3	3

8. LEARNING OUTCOMES :

- Understand the usage of alter table.

9. APPLICATION AREAS:

- Design and develop database applications for real world problems such as health care, education, industry, transport, supply chain etc.

10. REMARKS:

https://www.onlinegdb.com/online_sqlite_editor