F. Y. B. B. A. (C.A.) Semester I

Lab Book

Name: _____

College Name: _____

Roll No. _____

Division: _____

Academic Year: _____

CA-106:
Computer Laboratory Based on CA-104

(DBMS)

Assignment Completion Sheet

Index

Sr.No	Title	Signature
1	DDL Commands (Table Creation).	
2	DDL Commands(Alter and Drop table).	
3	DML Commands(Insert, Update and Delete).	
4	RDB without Constraints.	
5	Table Creation with Constraints.	
6	RDB with constraints.	
7	Implementation of Select Command	
8	SQL Set operation.	
9	Joins	
10	Case Study	

Name and Signature of Subject Teacher Head of Department

Date:-

Introduction

About the Book

This workbook is intended to be used by FYBBA (CA) students for their practical purpose. It helps to the students for clearing their theoretical as well as practical concepts.

Instructions to the Students and Instructors:

- Students should carry workbook while coming to the practical.
- Students should complete all the practical assignments within given time interval.
- Instructors should check all the assignments regularly and guide to the students accordingly.

Editors:

1. Mr. Satyavan M. Kunjir
2.Mr. Yogesh Ingale
3.Mrs. Malati Tribhuwan

Relational Model

❖ What is Relational Model?

The relational model represents the database as a collection of relations. A relation is nothing but a table of values. Every row in the table represents a collection of related data values. These rows in the table denote a real-world entity or relationship.

The table name and column names are helpful to interpret the meaning of values in each row. The data are represented as a set of relations. In the relational model, data are stored as tables. However, the physical storage of the data is independent of the way the data are logically organized.

❖ Some popular Relational Database Management Systems are:

- DB2 and Informix Dynamic Server – IBM
- Oracle and RDB – Oracle
- SQL Server and Access - Microsoft

❖ Relational Model Concepts:

1. **Attribute:** Each column in a Table. Attributes are the properties which define a relation. e.g., Student_Rollno, NAME,etc.
2. **Tables** – In the Relational model the, relations are saved in the table format. It is stored along with its entities. A table has two properties rows and columns. Rows represent records and columns represent attributes.
3. **Tuple** – It is nothing but a single row of a table, which contains a single record.
4. **Relation Schema:** A relation schema represents the name of the relation with its attributes.
5. **Degree:** The total number of attributes which in the relation is called the degree of the relation.
6. **Cardinality:** Total number of rows present in the Table.
7. **Column:** The column represents the set of values for a specific attribute.
8. **Relation instance** – Relation instance is a finite set of tuples in the RDBMS system. Relation instances never have duplicate tuples.
9. **Relation key** - Every row has one, two or multiple attributes, which is called relation key.
10. **Attribute domain** – Every attribute has some pre-defined value and scope which is known as attribute domain

❖ Relational Integrity constraints:

Relational Integrity constraints are referred to conditions which must be present for a valid relation. These integrity constraints are derived from the rules in the mini-world that the database represents. There are many types of integrity constraints. Constraints on the Relational database management system are mostly divided into three main categories are:

1. Domain constraints
2. Key constraints
3. Referential integrity constraints

❖ **Domain Constraints:**

Domain constraints can be violated if an attribute value is not appearing in the corresponding domain or it is not of the appropriate data type.

Domain constraints specify that within each tuple, and the value of each attribute must be unique. This is specified as data types which include standard data type's integers, real numbers, characters, Booleans, variable length strings, etc.

❖ **Key constraints:**

An attribute that can uniquely identify a tuple in a relation is called the key of the table. The value of the attribute for different tuples in the relation has to be unique.

Example:

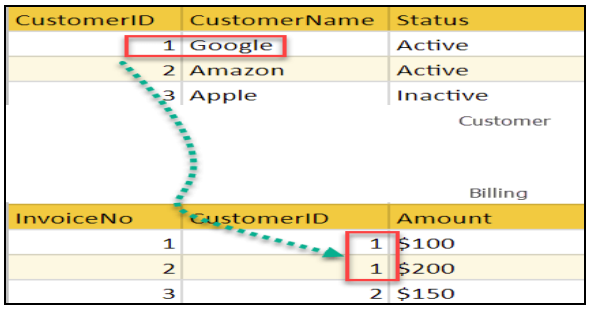
In the given table, CustomerID is a key attribute of Customer Table. It is most likely to have a single key for one customer, CustomerID =1 is only for the CustomerName =" Google".

CustomerID	CustomerName	Status
1	Google	Active
2	Amazon	Active
3	Apple	Inactive

❖ **Referential integrity constraints:**

Referential integrity constraints are base on the concept of Foreign Keys. A foreign key is an important attribute of a relation which should be referred to in other relationships. Referential integrity constraint state happens where relation refers to a key attribute of a different or same relation. However, that key element must exist in the table.

Example:



In the above example, we have 2 relations, Customer and Billing.

Tuple for CustomerID =1 is referenced twice in the relation Billing. So we know CustomerName=Google has billing amount $300

❖ **Operations in Relational Model:**

Four basic update operations performed on relational database model are
Insert, update, delete and select.

- Insert is used to insert data into the relation
- Delete is used to delete tuples from the table.
- Modify allows you to change the values of some attributes in existing tuples.
- Select allows you to choose a specific range of data.

Whenever one of these operations are applied, integrity constraints specified on the relational database schema must never be violated.

❖ **DBMS Keys: Primary, Candidate, Super, Alternate and Foreign (Example)**

❖ **What are Keys**?

A DBMS key is an attribute or set of an attribute which helps you to identify a row(tuple) in a relation(table). They allow you to find the relation between two tables. Keys help you uniquely identify a row in a table by a combination of one or more columns in that table.

Example:

Employee ID	FirstName	LastName
11	Andrew	Johnson
22	Tom	Wood
33	Alex	Hale

In the above-given example, employee ID is a primary key because it uniquely identifies an employee record. In this table, no other employee can have the same employee ID.

❖ **Why we need a Key?**

Here, are reasons for using Keys in the DBMS system.

- Keys help you to identify any row of data in a table. In a real-world application, a table could contain thousands of records. Moreover, the records could be duplicated. Keys ensure that you can uniquely identify a table record despite these challenges.
- Allows you to establish a relationship between and identify the relation between tables
- Help you to enforce identity and integrity in the relationship.

DBMS has following seven types of Keys each have their different functionality:

➢ **What is the Super key?**

A superkey is a group of single or multiple keys which identifies rows in a table. A Super key may have additional attributes that are not needed for unique identification.

Example:

EmpSSN	EmpNum	Empname
9812345098	AB05	Shown
9876512345	AB06	Roslyn
199937890	AB07	James

In the above-given example, EmpSSN and EmpNum name are superkeys.

- **What is a Primary Key?**

A column or group of columns in a table which helps us to uniquely identifies every row in that table is called a primary key. This DBMS can't be a duplicate. The same value can't appear more than once in the table.

- **Rules for defining Primary key:**

1. Two rows can't have the same primary key value
2. It must for every row to have a primary key value.
3. The primary key field cannot be null.
4. The value in a primary key column can never be modified or updated if any foreign key refers to that primary key.

Example:

In the following example, <code>StudID</code> is a Primary Key.

StudID	Roll No	First Name	LastName	Email
1	11	Tom	Price	abc@gmail.com
2	12	Nick	Wright	xyz@gmail.com
3	13	Dana	Natan	mno@yahoo.com

- **What is the Alternate key?**

All the keys which are not primary key are called an alternate key. It is a candidate key which is currently not the primary key. However, A table may have single or multiple choices for the primary key.

Example: In this table.

StudID, Roll No, Email are qualified to become a primary key. But since StudID is the primary key, Roll No, Email becomes the alternative key.

StudID	Roll No	First Name	LastName	Email
1	11	Tom	Price	abc@gmail.com
2	12	Nick	Wright	xyz@gmail.com
3	13	Dana	Natan	mno@yahoo.com

- **What is a Candidate Key?**

A super key with no repeated attribute is called candidate key.

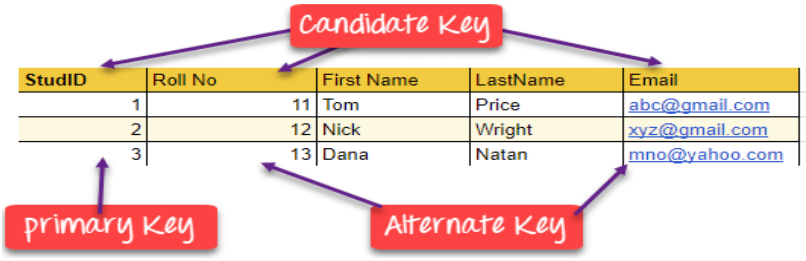
The Primary key should be selected from the candidate keys. Every table must have at least a single candidate key.

- **Properties of Candidate key:**

- It must contain unique values
- Candidate key may have multiple attributes
- Must not contain null values
- It should contain minimum fields to ensure uniqueness
- Uniquely identify each record in a table

Example: In the given table Stud ID, Roll No, and email are candidate keys which help us to uniquely identify the student record in the table.

StudID	Roll No	First Name	LastName	Email
1	11	Tom	Price	abc@gmail.com
2	12	Nick	Wright	xyz@gmail.com
3	13	Dana	Natan	mno@yahoo.com



What is the foreign key?

A foreign key is a column which is added to create a relationship with another table. Foreign keys help us to maintain data integrity and also allows navigation between two different instances of an entity. Every relationship in the model needs to be supported by a foreign key.

Example:

DeptCode	DeptName
001	Science
002	English
005	Computer

Teacher ID	Fname	Lname
B002	David	Warner
B017	Sara	Joseph
B009	Mike	Brunton

In this example, we have two table, teach and department in a school. However, there is no way to see which search work in which department.

In this table, adding the foreign key in Deptcode to the Teacher name, we can create a relationship between the two tables.

Teacher ID	DeptCode	Fname	Lname
B002	002	David	Warner
B017	002	Sara	Joseph
B009	001	Mike	Brunton

This concept is also known as Referential Integrity.

- **What is the Compound key?**

Compound key has many fields which allow you to uniquely recognize a specific record. It is possible that each column may be not unique by itself within the database. However, when combined with the other column or columns the combination of composite keys become unique.

Example:

OrderNo	PorductID	Product Name	Quantity
B005	JAP102459	Mouse	5
B005	DKT321573	USB	10
B005	OMG446789	LCD Monitor	20
B004	DKT321573	USB	15
B002	OMG446789	Laser Printer	3

In this example, OrderNo and ProductID can't be a primary key as it does not uniquely identify a record. However, a compound key of Order ID and Product ID could be used as it uniquely identified each record.

- **What is the Composite key?**

A key which has multiple attributes to uniquely identify rows in a table is called a composite key. The difference between compound and the composite key is that any part of the compound key can be a foreign key, but the composite key may or maybe not a part of the foreign key.

- **What is a Surrogate Key?**

An artificial key which aims to uniquely identify each record is called a surrogate key. These kind of key are unique because they are created when you don't have any natural primary key. They do not lend any meaning to the data in the table. Surrogate key is usually an integer.

Fname	Lastname	Start Time	End Time
Anne	Smith	09:00	18:00
Jack	Francis	08:00	17:00
Anna	McLean	11:00	20:00
Shown	Willam	14:00	23:00

Above, given example, shown shift timings of the different employee. In this example, a surrogate key is needed to uniquely identify each employee.

Surrogate keys are allowed when

- No property has the parameter of the primary key.
- In the table when the primary key is too big or complicated.

- **Difference between Primary key & foreign key:**

Primary Key	Foreign Key
Helps you to uniquely identify a record in the table.	It is a field in the table that is the primary key of another table.
Primary Key never accepts null values.	A foreign key may accept multiple null values.
Primary key is a clustered index and data in the DBMS table are physically organized in the sequence of the clustered index.	A foreign key cannot automatically create an index, clustered or non-clustered. However, you can manually create an index on the foreign key.
You can have the single Primary key in a table.	You can have multiple foreign keys in a table.

SQL (Structured Query Language)

- **What is SQL?**

Structured Query language (SQL) pronounced as "S-Q-L" or sometimes as "See-Quel" is the standard language for dealing with Relational Databases. A relational database defines relationships in the form of tables.

SQL programming can be effectively used to insert, search, update, delete database records.

That doesn't mean SQL cannot do things beyond that. It can do a lot of things including, but not limited to, optimizing and maintenance of databases.

Relational databases like MySQL Database, Oracle, Ms SQL Server, Sybase, etc. use SQL.

- **What is NoSQL?**

NoSQL is a non-relational DBMS, that does not require a fixed schema, avoids joins, and is easy to scale. NoSQL database is used for distributed data stores with humongous data storage needs. NoSQL is used for Big data and real-time web apps. For example companies like Twitter, Facebook, Google that collect terabytes of user data every single day.

NoSQL database stands for "Not Only SQL" or "Not SQL." Though a better term would NoREL NoSQL caught on. Carl Strozz introduced the NoSQL concept in 1998.

Traditional RDBMS uses SQL syntax to store and retrieve data for further insights. Instead, a NoSQL database system encompasses a wide range of database technologies that can store structured, semi-structured, unstructured and polymorphic data.

- **Difference between SQL and NoSQL**

Parameter	SQL	NOSQL
Definition	SQL databases are primarily called RDBMS or Relational Databases	NoSQL databases are primarily called as Non-relational or distributed database
Design for	Traditional RDBMS uses SQL syntax and queries to analyze and get the data for further insights. They are used for OLAP systems.	NoSQL database system consists of various kinds of database technologies. These databases were developed in response to the demands presented for the development of the modern application.
Query Language	Structured query language (SQL)	No declarative query language

Type	SQL databases are table based databases	NoSQL databases can be document based, key-value pairs, graph databases
Schema	SQL databases have a predefined schema	NoSQL databases use dynamic schema for unstructured data.
Ability to scale	SQL databases are vertically scalable	NoSQL databases are horizontally scalable
Examples	Oracle, Postgres, and MS-SQL.	MongoDB, Redis, , Neo4j, Cassandra, Hbase.
Best suited for	An ideal choice for the complex query intensive environment.	It is not good fit complex queries.
Hierarchical data storage	SQL databases are not suitable for hierarchical data storage.	More suitable for the hierarchical data store as it supports key-value pair method.
Variations	One type with minor variations.	Many different types which include key-value stores, document databases, and graph databases.
Development Year	It was developed in the 1970s to deal with issues with flat file storage	Developed in the late 2000s to overcome issues and limitations of SQL databases.
Open-source	A mix of open-source like Postgres & MySQL, and commercial like Oracle Database.	Open-source
Consistency	It should be configured for strong consistency.	It depends on DBMS as some offers strong consistency like MongoDB, whereas others offer only offers eventual consistency, like Cassandra.
Best Used for	RDBMS database is the right option for solving ACID problems.	NoSQL is a best used for solving data availability problems
Importance	It should be used when data validity is super important	Use when it's more important to have fast data than correct data
Best option	When you need to support dynamic queries	Use when you need to scale based on changing requirements
Hardware	Specialized DB hardware (Oracle Exadata, etc.)	Commodity hardware
Network	Highly available network (Infiniband,	Commodity network (Ethernet, etc.)

	Fabric Path, etc.)	
Storage Type	Highly Available Storage (SAN, RAID, etc.)	Commodity drives storage (standard HDDs, JBOD)
Best features	Cross-platform support, Secure and free	Easy to use, High performance, and Flexible tool.
Top Companies Using	Hootsuite, CircleCI, Gauges	Airbnb, Uber, Kickstarter
Average salary	The average salary for any professional SQL Developer is $84,328 per year in the U.S.A.	The average salary for "NoSQL developer" ranges from approximately $72,174 per year
ACID vs. BASE Model	ACID(Atomicity, Consistency, Isolation, and Durability) is a standard for RDBMS	Base (Basically Available, Soft state, Eventually Consistent) is a model of many NoSQL systems

- **When to use SQL?**

- SQL is the easiest language used to communicate with the RDBMS
- Analyzing behavioral related and customized sessions
- Building custom dashboards
- It allows you to store and gets data from the database quickly
- Preferred when you want to use joins and execute complex queries

- **When to use NoSQL?**

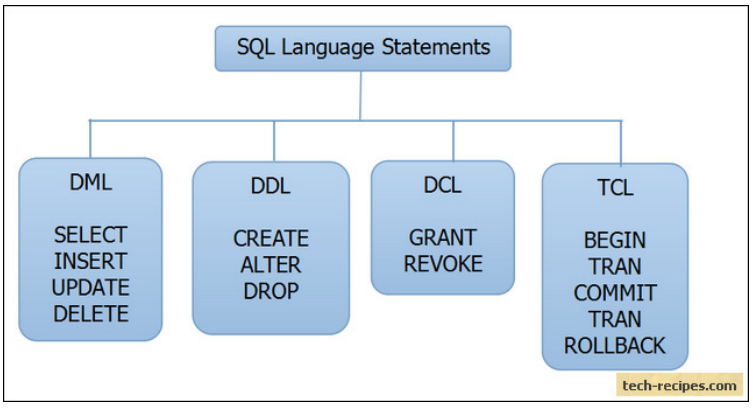
- **NoSQL DB (mongo) Vs RDBMS DB (mysql) Google Trend**

- When ACID support is not needed
- When Traditional RDBMS model is not enough
- Data which need a flexible schema
- Constraints and validations logic not required to be implemented in database
- Logging data from distributed sources
- It should be used to store temporary data like shopping carts, wish list and session data

SQL language is divided into four types of primary language statements: DML, DDL, DCL and TCL. Using these statements, we can define the structure of a database by creating and altering database objects, and we can manipulate data in a table through updates or deletions. We also can control which user can read/write data or manage transactions to create a single unit of work.

The four main categories of SQL statements are as follows:

1. DML (Data Manipulation Language)
2. DDL (Data Definition Language)
3. DCL (Data Control Language)
4. TCL (Transaction Control Language)



- **DML (Data Manipulation Language):**

DML statements affect records in a table. These are basic operations we perform on data such as selecting a few records from a table, inserting new records, deleting unnecessary records, and updating/modifying existing records.

- **DML statements include the following:**

SELECT – select records from a table
INSERT – insert new records

UPDATE – update/Modify existing records
DELETE – delete existing records

1. **Insert**: - Insert data into a table.

Syntax:-

```
INSERT INTO table_name (column, column1, column2, column3, ...)
VALUES (value, value1, value2, value3 ...)
```

Example:-

```
INSERT INTO Student (Roll_No, Name, Age) VALUES ('5','Satyavan','19');
```

2. **Update :-**

The UPDATE statement in SQL is used to update the data of an existing table in database. We can update single columns as well as multiple columns using UPDATE statement as per our requirement.

Syntax :

```
UPDATE table_name SET column1 = value1, column2 = value2,...
WHERE condition;

table_name: name of the table
column1: name of first , second, third column....
value1: new value for first, second, third column....
Condition: condition to select the rows for which the
           values of columns need to be updated.
```

Example:-

Updating single column: Update the column NAME and set the value to ‘Satyavan’ in all the rows where Age is 35.

```
UPDATE Student SET NAME = 'Satyavan' WHERE Age = 35;
```

3.DELETE Statement

The DELETE Statement in SQL is used to delete existing records from a table. We can delete a single record or multiple records depending on the condition we specify in the WHERE clause.

```
DELETE FROM table_name WHERE some_condition;

table_name: name of the table
some_condition: condition to choose particular record.
```

Example:-

```
DELETE FROM Student WHERE NAME = 'Satyavan';
```

- **DDL (Data Definition Language):**

DDL statements are used to alter/modify a database or table structure and schema. These statements handle the design and storage of database objects.

- CREATE – create a new Table, database, schema
- ALTER – alter existing table, column description
- DROP – delete existing objects from database.

- **The Create Command:**

The create table command defines each column of the table uniquely.
Each column has minimum of three attributes.

- Name
- Data type
- Size (column width).

Each table column definition is a single clause in the create table syntax.
Each table column definition is separated from the other by a comma.
Finally, the SQL statement is terminated with a semicolon.

Create Table "tablename"
("column1" "data type",
"column2" "data type",
"column3" "data type",
...
"ColumnN" "data type");

The Structure of Create Table Command
Table name is Student

Column name	Data type	Size
Roll_no	number	3
Name	Varchar2	30
DOB	Date	
Address	varchar2	50

Example:

```
CREATE TABLE Student
 (Roll_no number(3),
 Name varchar2(30),
 DOB date,
 Address varchar2(50));
```

- **The DROP Command:**

Syntax:
```
DROP TABLE <table_name>
```

Example:
```
DROP TABLE Student;
```
It will destroy the table and all data which will be recorded in it.

- **The ALTER Command:**

By The use of ALTER TABLE Command we can modify our exiting table.
Adding New Columns

Syntax:
```
ALTER TABLE <table_name>
ADD (<NewColumnName> <Data_Type>(<size>),.....n)
```

Example:
```
ALTER TABLE Student ADD (Age number(2), Marks number(3));
```

The Student table is already exist and then we added two more columns Age and Marks respectively, by the use of above command.

Dropping a Column from the Table

Syntax:
```
ALTER TABLE <table_name> DROP COLUMN <column_name>
```

Example:
```
ALTER TABLE Student DROP COLUMN Age;
```
This command will drop particular column

Modifying Existing Table

Syntax:
```
ALTER TABLE <table_name> MODIFY (<column_name> <NewDataType>(<NewSize>))
```

Example:
```
ALTER TABLE Student MODIFY (Name Varchar2(40));
```

The Name column already exist in Student table, it was char and size 30, now it is modified by Varchar2 and size 40.

- **Restriction on the ALTER TABLE:**

Using the ALTER TABLE clause the following tasks cannot be performed.

- Change the name of the table
- Change the name of the column

Decrease the size of a column if table data exists

- **DCL (Data Control Language):**

DCL statements control the level of access that users have on database objects.

- GRANT – allows users to read/write on certain database objects
- REVOKE – keeps users from read/write permission on database objects

- **TCL (Transaction Control Language):**

TCL statements allow you to control and manage transactions to maintain the integrity of data within SQL statements.

- BEGIN Transaction – opens a transaction
- COMMIT Transaction – commits a transaction
- ROLLBACK Transaction – ROLLBACK a transaction in case of any error

- **Fetching Data in the Table (Select Command)**

Once data has been inserted into a table, the next most logical operation would be to view what has been inserted. The SELECT SQL verb is used to achieve this.

All Rows and All Columns

Syntax: SELECT * FROM Table_name;

eg: Select * from Student;
It will show all the table records.

SELECT First_name, DOB FROM STUDENT WHERE Roll_no = '101'; Cover it by single inverted comma if its datatype is varchar or char.

This Command will show one row. Because you have given condition for only one row and particular records. If condition which has given in WHERE Clause is true then records will be fetched otherwise it will show no records selected.

- **Eliminating Duplicates:**

A table could hold duplicate rows. In such a case, you can eliminate duplicates.
Syntax: SELECT DISTINCT col, col, .., FROM table_name;
eg : SELECT DISTINCT * FROM Student;

or : SELECT DISTINCT first_name, city, pincode FROM Student;

It scans through entire rows, and eliminates rows that have exactly the same contents in each column.

- **Sorting DATA:**

The Rows retrieved from the table will be sorted in either Ascending or Descending order depending on the condition specified in select statement, the Keyword has used ORDER BY.

SELECT * FROM Student
ORDER BY First_Name;

it will in show records as alphabetical order from A to Z ascending order. If you want Descending order means Z to A then used DESC Keyword at last.

eg : SELECT first_name, city,pincode FROM Student
ORDER BY First_name DESC;

Aggregate Functions are all about

- Performing calculations on multiple rows
- Of a single column of a table
- And returning a single value.

Aggregate functions namely;

1) COUNT
2) SUM
3) AVG
4) MIN
5) MAX

Why use aggregate functions.

From a business perspective, different organization levels have different information requirements. Top levels managers are usually interested in knowing whole figures and not necessary the individual details.

Aggregate functions allow us to easily produce summarized data from our database.

- Least rented movies.
- Most rented movies.

- Average number that each movie is rented out in a month.

We easily produce above reports using aggregate functions.

Aggregate functions in detail.

COUNT Function:

The COUNT function returns the total number of values in the specified field. It works on both numeric and non-numeric data types. All aggregate functions by default exclude nulls values before working on the data.

COUNT (*) is a special implementation of the COUNT function that returns the count of all the rows in a specified table. COUNT (*) also considers Nulls and duplicates.

The table shown below shows data in movierentals table

reference_ number	transaction_ date	return_date	membership_ number	movie_id	movie_ returned
11	20-06-2012	NULL	1	1	0
12	22-06-2012	25-06-2012	1	2	0
13	22-06-2012	25-06-2012	3	2	0
14	21-06-2012	24-06-2012	2	2	0
15	23-06-2012	NULL	3	3	0

Let's suppose that we want to get the number of times that the movie with id 2 has been rented out
```
SELECT COUNT(`movie_id`)  FROM `movierentals` WHERE `movie_id` = 2;
```

Executing the above query in SQL

COUNT('movie_id')
3

- **DISTINCT Keyword:**

The DISTINCT keyword that allows us to omit duplicates from our results. This is achieved by grouping similar values together.

To appreciate the concept of Distinct, lets execute a simple query

```
SELECT `movie_id` FROM `movierentals`;
```

movie_id
1
2
2
2
3

Now let's execute the same query with the distinct keyword -

```
SELECT DISTINCT `movie_id` FROM `movierentals`;
```

As shown below, distinct omits duplicate records from the results.

movie_id
1
2
3

MIN function:

The MIN function returns the smallest value in the specified table field.

As an example, let's suppose we want to know the year in which the oldest movie in our library was released, we can use MIN function to get the desired information.

The following query helps us achieve that

```
SELECT MIN(`year_released`) FROM `movies`;
```

Executing the above query in SQL

MIN('year_released')
2005

MAX function:

Just as the name suggests, the MAX function is the opposite of the MIN function. It returns the largest value from the specified table field.

Let's assume we want to get the year that the latest movie in our database was released. We can easily use the MAX function to achieve that.

The following example returns the latest movie year released.

```
SELECT MAX(`year_released`)  FROM `movies`;
```

MAX('year_released')
2012

SUM function:

Suppose we want a report that gives total amount of payments made so far. We can use the SUM function which returns the sum of all the values in the specified column. SUM works on numeric fields only. Null values are excluded from the result returned.

The following table shows the data in payments table-

payment_ id	membership_ number	payment_ date	Description	amount_ paid	external_ reference _number
1	1	23-07-2012	Movie rental payment	2500	11
2	1	25-07-2012	Movie rental payment	2000	12
3	3	30-07-2012	Movie rental payment	6000	NULL

The query shown below gets the all payments made and sum them up to return a single result.

```
SELECT SUM(`amount_paid`) FROM `payments`;
```

SUM('amount_paid')
10500

AVG function:

AVG function returns the average of the values in a specified column. Just like the SUM function, it works only on numeric data types.

Suppose we want to find the average amount paid. We can use the following query -

```
SELECT AVG(`amount_paid`)  FROM `payments`;
```

AVG('amount_paid')
3500

Normalization

Normalization is a process of organizing the data in database to avoid data redundancy, insertion anomaly, update anomaly & deletion anomaly. Let's discuss about anomalies first then we will discuss normal forms with examples.

- **Anomalies in DBMS**

There are three types of anomalies that occur when the database is not normalized. These are – Insertion, update and deletion anomaly. Let's take an example to understand this.

Example: Suppose a manufacturing company stores the employee details in a table named employee that has four attributes: emp_id for storing employee's id, emp_name for storing employee's name, emp_address for storing employee's address and emp_dept for storing the department details in which the employee works. At some point of time the table looks like this:

emp_id	emp_name	emp_address	emp_dept
101	Rick	Delhi	D001
101	Rick	Delhi	D002
123	Maggie	Agra	D890
166	Glenn	Chennai	D900

The above table is not normalized. We will see the problems that we face when a table is not normalized.

Update anomaly: In the above table we have two rows for employee Rick as he belongs to two departments of the company. If we want to update the address of Rick then we have to update the same in two rows or the data will become inconsistent. If somehow, the correct address gets updated in one department but not in other then as per the database, Rick would be having two different addresses, which is not correct and would lead to inconsistent data.

- **Insert anomaly**: Suppose a new employee joins the company, who is under training and currently not assigned to any department then we would not be able to insert the data into the table if emp_dept field doesn't allow nulls.

- **Delete anomaly**: Suppose, if at a point of time the company closes the department D890 then deleting the rows that are having emp_dept as D890 would also delete the information of employee Maggie since she is assigned only to this department.

To overcome these anomalies we need to normalize the data. In the next section we will discuss about normalization.

❖ **Normalization**

Here are the most commonly used normal forms:

- First normal form(1NF)
- Second normal form(2NF)
- Third normal form(3NF)
- Boyce & Codd normal form (BCNF)

➢ **First normal form (1NF):**

As per the rule of first normal form, an attribute (column) of a table cannot hold multiple values. It should hold only atomic values.

Example: Suppose a company wants to store the names and contact details of its employees. It creates a table that looks like this:

Emp_id	Emp_name	Emp_address	Emp_mobile
101	Herschel	New Delhi	8912312390
102	Jon	Kanpur	8812121212 9900012222
103	Ron	Chennai	7778881212
104	Lester	Bangalore	9990000123 8123450987

Two employees (Jon & Lester) are having two mobile numbers so the company stored them in the same field as you can see in the table above.

This table is not in 1NF as the rule says "each attribute of a table must have atomic (single) values", the emp_mobile values for employees Jon & Lester violates that rule.

To make the table complies with 1NF we should have the data like this:

Emp_id	Emp_name	Emp_address	Emp_mobile
101	Herschel	New Delhi	8912312390
102	Jon	Kanpur	8812121212
102	Jon	Kanpur	9900012222
103	Ron	Chennai	7778881212
104	Lester	Bangalore	9990000123
104	Lester	Bangalore	8123450987

- **Second normal form (2NF):**

A table is said to be in 2NF if both the following conditions hold:

- Table is in 1NF (First normal form)
- No non-prime attribute is dependent on the proper subset of any candidate key of table.

An attribute that is not part of any candidate key is known as non-prime attribute.

Example: Suppose a school wants to store the data of teachers and the subjects they teach. They create a table that looks like this: Since a teacher can teach more than one subjects, the table can have multiple rows for a same teacher.

Teacher_id	**Subject**	**Teacher_age**
111	Maths	38
111	Physics	38
222	Biology	38
333	Physics	40
333	Chemistry	40

Candidate Keys: {teacher_id, subject}
Non prime attribute: teacher_age

The table is in 1 NF because each attribute has atomic values. However, it is not in 2NF because non prime attribute teacher_age is dependent on teacher_id alone which is a proper subset of candidate key. This violates the rule for 2NF as the rule says “no non-prime attribute is dependent on the proper subset of any candidate key of the table”.

To make the table complies with 2NF we can break it in two tables like this: **teacher_details table:**

Teacher_id	Teacher_age
111	38
222	38
333	40

teacher_subject table:

teacher_id	Subject
111	Maths
111	Physics
222	Biology
333	Physics
333	Chemistry

Now the tables comply with Second normal form (2NF).

- **Third Normal form (3NF)**

A table design is said to be in 3NF if both the following conditions hold:

- Table must be in 2NF
- Transitive functional dependency of non-prime attribute on any super key should be removed.

An attribute that is not part of any candidate key is known as non-prime attribute.

In other words 3NF can be explained like this: A table is in 3NF if it is in 2NF and for each functional dependency X-> Y at least one of the following conditions hold:

- X is a super key of table
- Y is a prime attribute of table

An attribute that is a part of one of the candidate keys is known as prime attribute.

Example: Suppose a company wants to store the complete address of each employee, they create a table named employee_details that looks like this:

emp_id	emp_name	emp_zip	emp_state	emp_city	emp_district
1001	John	282005	UP	Agra	Dayal Bagh
1002	Ajeet	222008	TN	Chennai	M-City
1006	Lora	282007	TN	Chennai	Urrapakkam
1101	Lilly	292008	UK	Pauri	Bhagwan
1201	Steve	222999	MP	Gwalior	Ratan

Super keys: {emp_id}, {emp_id, emp_name}, {emp_id, emp_name, emp_zip}…so on
Candidate Keys: {emp_id}
Non-prime attributes: all attributes except emp_id are non-prime as they are not part of any candidate keys.

Here, emp_state, emp_city & emp_district dependent on emp_zip. And, emp_zip is dependent on emp_id that makes non-prime attributes (emp_state, emp_city & emp_district) transitively dependent on super key (emp_id). This violates the rule of 3NF.

To make this table complies with 3NF we have to break the table into two tables to remove the transitive dependency:

employee table:

Emp_id	**Emp_name**	**Emp_zip**
1001	John	282005
1002	Ajeet	222008
1006	Lora	282007
1101	Lilly	292008
1201	Steve	222999

employee_zip table:

emp_zip	emp_state	emp_city	emp_district
282005	UP	Agra	Dayal Bagh
222008	TN	Chennai	M-City
282007	TN	Chennai	Urrapakkam
292008	UK	Pauri	Bhagwan
222999	MP	Gwalior	Ratan

- **Boyce Codd normal form (BCNF):**

It is an advance version of 3NF that's why it is also referred as 3.5NF. BCNF is stricter than 3NF. A table complies with BCNF if it is in 3NF and for every functional dependency X->Y, X should be the super key of the table.

Example: Suppose there is a company wherein employees work in more than one department. They store the data like this:

emp_id	emp_nationality	emp_dept	dept_type	dept_no_of_emp
1001	Austrian	Production and planning	D001	200
1001	Austrian	Stores	D001	250
1002	American	design and technical support	D134	100
1002	American	Purchasing department	D134	600

Functional dependencies in the table above:
emp_id->emp_nationality
emp_dept -> {dept_type, dept_no_of_emp}

Candidate key: {emp_id, emp_dept}

The table is not in BCNF as neither emp_id nor emp_dept alone are keys.

To make the table comply with BCNF we can break the table in three tables like this:
emp_nationality table:

Emp_id	**Emp_nationality**
1001	Austrian
1002	American

emp_dept table:

Emp_dept	**Dept_type**	**Dept_no_of_emp**
Production and planning	D001	200
Stores	D001	250
design and technical support	D134	100
Purchasing department	D134	600

emp_dept_mapping table:

Emp_id	Emp_dept
1001	Production and planning
1001	Stores
1002	design and technical support
1002	Purchasing department

Functional dependencies:
emp_id -> emp_nationality
emp_dept -> {dept_type, dept_no_of_emp}
Candidate keys:
For first table: emp_id
For second table: emp_dept
For third table: {emp_id, emp_dept}

This is now in BCNF as in both the functional dependencies left side part is a key.

Exercise 1: DDL(Data Definition Language)

Creating Tables:

- **Create table for the information given below by choosing appropriate data types and also specifying proper primary key constraint on fields which are underlined**

1. Player (player_id , name, Birth_date ,Birth_place, game_name)
2. Student (roll_no, name,class,per,birth_date)
3. Project (project_id, project_name , project_description ,status)
4. Donor (donor_no, donor_name,blood_group,last_date)

- **Create table for the information given below by choosing appropriate data types and also specifying proper primary key constraint on fields which are underlined.**

1. Property (property_id, property_desc , area, rate, agri_status)
2. Actor (actor_id, Actor_name, birth_date)
3. Movie(movie-no, name, release-year)
4. Hospital(hno,hname,hcity)

- **Create table for the information given below by choosing appropriate data types and also specifying proper primary key constraint on fields which are underlined.**

1. Employee(ENo, EName, Joining_date,company_name,salary,Designation)
2. College(College_Code,College_Name,Address,Establish_year)
3. Doctor(Dno, Dname, Specialization,Qualification)
4. ClassRoom(CRoomNo,location,capacity)

Signature of the instructor | Date | Remark

Exercise 2: Alter Table and Drop Table

- Create table student(Roll_no, sname, date_of_birth). Add new column into student relation named address as a text data type and a column phone of data type integer.

- Create table driver (licence_no, Name, Address) and perform the following queries
    1. Add new column age of data type integer.
    2. Alter table by modifying driver_name to "Patil"
    3. Alter table driver ,drop the column age.
    4. Remove the driver table from the database.

- Create table Game (name, no-of-players, captain_name) and perform the following queries
    1. Add new column game_no of data type integer.
    2. Alter table by adding constraint uppercase to captain_name.
    3. Modify table by adding the column game_duration.
    4. Add column game_type with values cricket,hockey,tennis.
    5. Remove game table from the database.

Signature of the instructor | Date | Remark

Exercise 3: DML Commands

- **Consider the following table Employee(ENo, EName, Salary, DOJ,Qualification) and answer the following query.**
    1. Insert at least five records into the table.
    2. Update the salary of employee to 50000 whose ENo is 1.
    3. Delete the details of employee whose ENo is 5.
    4. Update the Qualification of employee to "MCS NET" whose Name is Mr.Satyavan.
    5. Update the salary of employee to 40000 whose qualification is "MCS NET" and Name is "Ajay"

- **Consider the following table Hospital (HNo, HName, Addr, Est_Year , speciality) and answer the following query.**
    1. Insert at least five records into the table.
    2. Update an address of hospital to "Pimple Gurav" whose name is "Birla".
    3. Update the specialty of hospital to "Multi" whose established year is between 1990 to 2000.
    4. Delete the details of Hospital whose address is "Pimpri".

- **Consider the following table Student (Roll_No, Name, class, DOB, college) and answer the following query.**
    1. Insert at least 10 records into the table.
    2. Update the class of student to "TY" whose birth date is '18/03/1999'.
    3. Delete the details of students whose college is "Dr.D Y Patil".
    4. Update the college of student to "Dr. D Y Patil " whose name is "Yash".

Signature of the instructor | Date | Remark

Exercise 4: RDB without Constraints:

- **Consider the following entities and their relationships. Create a RDB in 3 NF for the following and answer the queries:**

  Emp(eno ,ename ,designation ,salary,DOJ)
  Dept(dno,dname ,loc)

  The relationship between Dept & Emp is one-to-many.

  1. Insert at least five records into the tables.
  2. Display the names of employees who are working in "Quality Department".
  3. Display the name of employee who is 'Manager' of "Purchase Department".
  4. Display the name of department whose location is "Baramati" and "Mr. Pawar" is working in it.
  5. Display the names of employees whose salary is greater than 50000 and department is "Quality".

- **Consider the following entities and their relationships. Create a RDB in 3 NF for the following and answer the queries:**

  Hospital(hno ,hname , city, Est_year)
  Doctor(dno , dname , addr, Speciality)

  The relationship between Hospital and Doctor is one - to – Many

  1. Insert at least 10 records into the tables.
  2. Display the names of hospitals which are located at "Pimpri" city.
  3. Display the names of doctors who are working in "Birla" Hospital and city name is "Chinchwad".
  4. Display the specialty and name of doctor who is working in "Ruby" hospital and his address is "Pimple Gurav".
  5. Display the names of doctors whose speciality is "medicine"

- **Consider the following entities and their relationships. Create a RDB in 3 NF for the following and answer the queries:**

  Patient (PCode, PName , Addr , Disease)
  Bed (Bed_No, RoomNo, loc)

  Relationship: - A one-one relationship between patient and bed.

  1. Insert at least five records into the tables.
  2. Display the names of patients who are admitted in room no 101.
  3. Display the disease of patient whose bed_No is 1.
  4. Give the roon_no and bed_no of patient whose name is "Mr Ajay".

Signature of the instructor | Date | Remark

Exercise 5: Table Creation with Constraints:

- **Consider the following tables and integrity constraints given and create the tables accordingly:**

1. **Machine**(Mid, MName NOT NULL, MType, MPrice , MCost)

   Constraints: 1. MName should be in uppercase.

   2. MType can be ('drilling', 'milling', 'lathe', 'turning', 'grinding').
   3. MPrice should be greater than zero.

   Table level constraint: MCost less than MPrice.

2. **Policy**(No, Name NOT NULL, Type , Sale_Date, Intro_date)

   Constraints: 1. Name should be in lowercase.
   2. Type can be ('life', 'vehicle', 'accident')

   Table level constraint: Sale_date should be greater than Intro_date.

3. **Employee** (EmpNo, Emp_Name NOT NULL, Emp_desig, Emp_sal , Emp_uid)

   Constraints:
   1. Emp_name should be in uppercase.
   2. Emp_desg can be ('Manager', 'staff', 'worker').
   3. Emp_sal should be greater than zero.

   Table level constraint: Emp_uid not equal to Emp_id

3 **Room**(room_no , type, price);

   Constraints:
   1. Room type must be one of single, double, family.
   2. Price must be between Rs.500/- and 1000/-.
   3. Room no must be between 1 and 100.

Signature of the instructor | Date | Remark

Exercise 6: RDB with Constraints:

- **Consider the following Entities and Relationships**

  Sales_order(ordNo, ordDate)
  Client (clientNo, ClientName, addr)

  Constraint: Primary key, ClientName should not be NULL.

  A client can give one or more sales_orders ,but a sales_order belongs to exactly one client. Create the relations accordingly, so that the relationship is handled properly and the relations are in normalized form(3 NF) and perform the following tasks.

  1. Insert two client records into client table.
  2. Insert 3 sales records for each client.
  3. Change order date of client_No 'C004' to 18/03/2019
  4. Delete all sale records having order date before 10 /02/2018.
  5. Display date wise sales_order given by clients.

- **Consider the following Entities and Relationships**

  Customer (cust_no, cust_name, address, city)
  Loan (loan_no, loan_amt)

  Relation between Customer and Loan is **Many to Many**

  Constraint: Primary key, loan_amt should be > 0.

  Create a Database in 3NF & write queries for following.

  1. Find details of all customers whose loan is greater than 10 lakhs.
  2. List all customers whose name starts with 'sa'.
  3. List names of all customers in descending order who has taken a loan in Pimpri city.
  4. Display customer details having maximum loan amount.
  5. Calculate total of all loan amount.

- **Consider the following Entities and Relationships**

  Department (dept_no, dept_name, location)
  Employee (emp_no, emp_name, address, salary, designation)
  Relation between Department and Employee is **One to Many**

  Constraint: Primary key, salary should be > 0.

  Create a Database in 3NF & write queries for following.

  1. Find total salary of all the employees from computer science dept.
  2. Find the name of department whose average salary is above 10000.
  3. Count the number of employees in each department.
  4. Display the maximum salary of each department.
  5. Display department wise employee list.
  6. Increase Salary of "Managers" by 15%
  7. Delete all Employees who are working as "clerk".

❖ **Consider the following Entities and Relationships**

Project (pno, pname, start_date, budget, status)
Department (dno, dname, HOD)
Relation between Project and Department is **Many to One**
Constraint: Primary key.
Project Status Constraints: C – completed,
P-Progressive, I-Incomplete

Create a Database in 3NF & write queries for following.

1. List the project name and department details worked in projects that are 'Complete'.
2. Display total budget of each department.
3. Display incomplete project of each department
4. Find the names of departments that have budget greater than 50000 .
5. Display all project working under 'Mr.Desai'.

❖ **Consider the following Entities and Relationships**

Room (roomno, desc, rate)
Guest (gno, gname, no_of_days)
Relation between Room and Guest is **One to One**.

Constraint: Primary key, no of days should be > 0.

Create a Database in 3NF & write queries for following.

1. Display room details according to its rates in ascending order.
2. Find the names of guest who has allocated room for more than 3 days.
3. Find no. of AC rooms.
4. Display total amount for NON-AC rooms.
5. Find names of guest with maximum room charges.

❖ **Consider the following Entities and Relationships**

Book (Book_no, title, author, price, year_published)
Customer (cid, cname, addr)
Relation between Book and Customer is **Many to Many** with quantity as descriptive attribute.

Constraint: Primary key, price should be >0.

Create a Database in 3NF & write queries for following.

1. Display customer details from 'Mumbai'.
2. Display author wise details of book.
3. Display all customers who have purchased the books published in the year 2013.
4. Display customer name that has purchased more than 3 books.
5. Display book names having price between 100 and 200 and published in the year 2013.

❖ **Consider the following Entities and Relationships**

Property (pno, desc, area, rate)
Owner (owner_name, addr, phno)
Relation between owner and Property is **One to Many**.

Constraint: Primary key, rate should be > 0

Create a Database in 3NF & write queries for following.

1. Display area wise property details.
2. Display property owned by 'Mr.Patil' having minimum rate.
3. Display all properties with owner name that having highest rate of properties located in Chinchwad area.
4. Display owner name having maximum no. of properties.
5. Delete all properties from "pune" owned by " Mr. Joshi".
6. Display all the properties from Mumbai owned by "Mr. Patil".
7. Update the phone Number of "Mr Talure" to 9923323366 who having property at Pimpri.

❖ **Consider the following Entities and Relationships**

Employee (emp_no, name, skill, payrate)
Position (posting_no, skill)
Relation between Employee and Position is **Many to Many** with day and shift as descriptive attribute.

Constraint: Primary key, payrate should be > 0.

Create a Database in 3NF & write queries for following.

1. Find the names and rate of pay all employees who allocated a duty.
2. Give employee number who are working at posting_no. 201, but don't have the skills of waiter.
3. Display a list of names of employees who have skill of chef and who has assigned a duty.
4. Display emp_no and dates for all employees who are working on Tuesday and at least one other day.
5. Display shiftwise employee details.

❖ **Consider the following Entities and Relationships**

Bill (billno, day, tableno, total)
Menu (dish_no, dish_desc, price)
Relation between Bill and Menu is **Many to Many** with quantity as descriptive attribute.

Constraint: Primary key, price should be > 0.

Create a Database in 3NF & write queries for following.

1. Display receipt which includes bill_no with Dish description, price, quantity and total amount of each menu.
2. Find total amount collected by hotel on date 08/01/2013
3. Count number of menus of billno 301.
4. Display menu details having price between 100 and 500.
5. Display total number of bills collected from each table on 01/12/2013.

Signature of the instructor | Date | Remark

Exercise 7: **Demonstration of Select Command**

- **Create the following tables (primary keys are underlined).**

Emp(eno,ename,sal,address,ph_no)
Dept(dno, name, loc)

Emp and Dept are related with many to one with each other. Create the Relations accordingly, so that the relationship is handled properly and relations are in normalized form (3NF).

Execute following select queries & write the business task performed by each query.

1. Select * from emp;
2. Select empno, name from emp;
3. Select distinct deptno from emp;
4. Select * from emp where deptno = ___;
5. Select * from emp where address = 'pune' and sal > _____;
6. Select * from emp where address = 'pune' and salary between _____ and _____;
7. Select * from emp where name like '---%'
8. Select * from emp where name like '%and%'
9. Select * from emp where salary is null;
10. Select * from emp order by eno;
11. Select * from emp order by deptno, eno desc;
12. Select deptno as department, sum(salary) as total from emp group by deptno order by deptno;
13. Select deptno as department , count(eno) as total_emp from emp group by deptno having count(eno) > _____ order by deptno;
14. select avg(salary) from emp;
15. select max(salary),deptno from emp group by deptno having max(sal) > _____;
16. select deptno, min(salary) from emp order by deptno;
17. update emp set salary = salary + 0.5*salary where deptno = (select deptno from department where dname = 'finance');
18. update emp set deptno = (select deptno from department where dname = 'finance') Where deptno = (select deptno from department where dname = 'inventory');

- **Create the following tables (primary keys are underlined).**

Person (pnumber, pname, birthdate, income)
Area(aname, area_type)

An area can have one or more person living in it , but a person belongs to exactly one area. The attribute 'area_type' can have values as either urban or rural.

Create the Relations accordingly, so that the relationship is handled properly and the relations are in normalized form (3NF).

Assume appropriate data types for all the attributes. Add any new attributes as required, depending on the queries. Insert sufficient number of records in the relations / tables with appropriate values as suggested by some of the queries.

Execute following select queries & write the business task performed by each query.

1 List the names of all people living in '_____' area.
2 List details of all people whose names start with the alphabet '_' & contains maximum _alphabets in it.
3 List the names of all people whose birthday falls in the month of _____.
4 Give the count of people who are born on '_____'
5 Give the count of people whose income is below _____.
2. List names of all people whose income is between _____ and _____;
3. List the names of people with average income
4. List the sum of incomes of people living in '____'
5. List the names of the areas having people with maximum income (duplicate areas must be omitted in the result)
6. Give the count of people in each area
7. List the details of people living in '____' and having income greater than _____;
8. List the details pf people, sorted by person number
9. List the details of people, sorted by area, person name
10. List the minimum income of people.
11. Transfer all people living in 'pune' to 'mumbai'.
12. Delete information of all people staying in 'urban' area

❖ **Create the following tables (primary keys are underlined).**
Emp (eno,name,dno,salary)
Project (pno,pname,control_dno,budget)

Each employee can work on one or more projects, and a project can have many employees working in it. The number of hours worked on each project by an employee also needs to be stored.

Create the Relations accordingly, so that the relationship is handled properly and the relations are in normalized form (3NF).

Assume appropriate data types for the attributes. Add any new attributes, new relations as required by the queries.

Insert sufficient number of records in the relations / tables with appropriate values as suggested by some of the queries.

Write the queries for following business tasks & execute them.

1. list the names of departments that controls projects whose budget is greater than ___
2. list the names of projects, controlled by department No __, whose budget is greater than atleast one project controlled by department No __
3. list the details of the projects with second maximum budget
4. list the details of the projects with third maximum budget.
5. list the names of employees, working on some projects that employee number __ is working.
6. list the names of employees who do not work on any project that employee number __ works on
7. list the names of employees who do not work on any project controlled by

'_____' department

8. list the names of projects along with the controlling department name, for those projects which has atleast __ employees working on it.
9. list the names of employees who is worked for more than 10 hrs on atleast one project controlled by '_____' dept.
10. list the names of employees , who are males , and earning the maximum salary in their department.
11. list the names of employees who work in the same department as '_____'.
12. list the names of employees who do not live in _____ or _____.

❖ **Create the following tables (primary keys are underlined).**
Movies (M_name,release_year,budget)
Actor (A_name,role,charges,A_address)
Producer (producer_id,name,P_address)

Each actor has acted in one or more movie. Each producer has produced many movies but each movie can be produced by more than one producers. Each movie has one or more actors acting in it, in different roles.

Create the Relations accordingly, so that the relationship is handled properly and the relations are in normalized form (3NF).

Assume appropriate data types for the attributes. Add any new attributes, new relations as required by the queries.

Insert sufficient number of records in the relations / tables with appropriate values as suggested by some of the queries.

Write the queries for following business tasks & execute them.

1. List the names of actors who have acted in at least one movie, in which 'shahrukh' has acted.
2. List the names of movies with the highest budget.
3. List the names of movies with the second highest budget
4. List the names of actors who have acted in the maximum number of movies.
5. List the names of movies, produced by more than one producer.
6. List the names of actors who are given with the maximum charges for their movie.
7. List the names of producers who produce the same movie as '_____'.
8. List the names of actors who do not live in _____ or _____.

Signature of the instructor | **Date** | **Remark**

Exercise 8: SQL Set operations

You can combine multiple queries using the set operators UNION, UNION ALL, INTERSECT and Except. ALL set operators have equal precedence.

1. Union: Returns the union of two sets of values, eliminating duplicates.

Syntax: <select query>
Union
<select query>

2. **Union all:** Returns the union of two sets of values, retaining all duplicates.

Syntax: <select query>
Union all
<select query>

3 **Intersect:** Returns the intersection of two sets of values, eliminating duplicates.

Syntax: <select query>
intersect
<select query>

4 **Intersect all:** Returns the intersection of two sets of values, retaining duplicates.

Syntax: <select query>
Intersect all
<select query>

5 **Except:** Returns the difference between two set of values, I.e returns all values from set1 , not contained in set2 .eliminates duplicates.

Syntax: <select query>
except
<select query>

6 **Except all:** Returns the difference between two set of values, i.e. returns all values from set1, Not contained in set2 .Retains all duplicates.

Syntax: <select query>
Except all
<select query>

- **Create the following tables. (Primary Keys are underlined)**

  Emp(emp_id ,emp_name, address, bdate)
  Investor (inv_name , inv_no, inv_date, inv_amt)

  An employee may invest in one or more investments; hence he can be an investor. But an investor need not be an employee of the firm.

  Create the Relations accordingly, so that the relationship is handled properly and the relations are in normalized form (3NF).

  Assume appropriate data types for the attributes. Add any new attributes, as required by the Queries. Insert sufficient number of records in the relations / tables.

Write the following queries & execute them.

1. List the distinct names of customers who are either employees, or investors or both.
2. List the names of customers who are either employees, or investors or both.
3. List the names of employees who are also investors.
4. List the names of employees who are not investors.

❖ **Create the following tables. (Primary Keys are underlined)**

Student (rno,sname,address,class)
Subject (subno,subname)

Student and Subject are related with many-to-many relationship with attribute marks and status. Create the Relations accordingly, so that the relationship is handled properly and the relations are in normalized form (3NF).

Write the following queries & execute them.

1. List the distinct names of students who have either Electronics, or Statistics or both subjects.
2. List the names of students who are either passed or failed.
3. List the students who have "Database" subject and they are not in "TY" class.
4. List the names of students who are not failed in any subject.
5. List the names of students not staying at "Uruli Kanchan".

Signature of the instructor | **Date** | **Remark**

Exercise 9: Joins

❖ **Consider the following relations to understand the use of joins.**

Student (s_id , sname, level ,age , subject)
Class (cname , meetat ,room, fid)
Enrolled (s_id i, cname)
Faculty (fid ,fname ,deptid)

The meaning of above relationship is enrolled has one record per student _class pair such that the student is enrolled in the class. Read the query carefully and insert sufficient number of records in the relations / tables with appropriate values to perform the following queries.

1. **Find the names of all classes that either meet in room R128 or have five or more students enrolled.**
   Sql>Select c.name from class c where c.room =’r128’ or c.name in (select e.cname from enrolled e group by e.name having count(*)>= 5);

2. **Find the name of the oldest student who is either a history subject or enrolled in a course taught by I.teach.**
   Sql> Select max(s.age) from student s where (s.subject=’history’) or s.num in (select e.num from class c ,enrolled e ,faculty f where e.name =c.name and c.fid=f.id and f.fname =’I.teach);

3. **Find the names of students enrolled in the maximum number of classes.**
   Sql> Select distinct s.name from student s where s.num in (select e.num from enrolled e group by e.num having count(*) >=all (select count(*) from enrolled e2 group by e2.num));

4. **Find the names of student not enrolled in any class.**
   Sql> Select distinct s.name from student s where s.num not in (select e.num from enrolled e);

5. **Find the names of faculty members who teach in every room in which some class is taught.**
   Sql> Select distinct f.name from faculty f where not exists ((select * from class) except (select c1.room from class c1 where c1.fid = f.fid));

Signature of the instructor | **Date** | **Remark**

Exercise 10: Case Study

- **Consider the following case study:**

A housing society needs to manage the administrative information related to the society. The society is made up of different types of flats like 2BHK, 1BHK, 3BHK. Each type has a well defined square-feet area . The outright sale rate & the rental value of the flat depends on the type of the flat. Each flat has a single owner. Each owner can have one or more flats in his name. The name, address , phone etc of the owner need to be maintained. For each flat, its type, the floor no, any internal specifications needs to be maintained.

The society also contains a club-house, which is rented out to flat owners , at a nominal rate for conducting various functions / programmes. Society would like to print reports like number of functions held in the club-house during a month / period etc.

Every month maintenance amount is collected from the owners of the flats. Society needs to maintain this finance information, like how much amount collected for a month, whether any defaulters for a month, sending reminders to the defaulters etc. The expenditure information includes money spent on maintenance of the society like paying the sweepers, cleaners of the common area of the society, any emergency expense, salaries of the security etc. Every month the society would like to print a report of expenditure versus collection.

Design the relational database for the above, so that the following queries can be answered:

1. List the flats of 2bhk type.
2. List the 3bhk flats that are currently vacant.
3. List the functions held in clubhouse during the month of "_____"
4. List the names of owners , who have never conducted any functions in the clubhouse.
5. List the payment defaulters for the month of "April"
6. List the total expenditure for the month of _____
7. List the month with the least expenditure.
8. Transfer the flat in the name of _____ to _____
9. List the names of owners, who own both a 2bhk and a _____

- **Consider the following case study :**

A 4-wheeler rental company needs to develop a database to store the following information : the information about the cars , like the registration number, the chassis number, the type of the vehicle (car, jeep, SUV etc). The vehicles may have one or more luxurious features like AC, Stereo, tape, DVD player etc).

The company also needs to maintain the information about its drivers like driver license no, name, address , age etc.

A car is driven by different drivers on different days, a driver may drive different cars on different days. The company also needs information regarding the different places to which the car had been driven down, the names of drivers who have driven it to these places along with the name of customers who had booked the car to that place. The information of the different destinations to which the cars from this company can be driven down, also needs to be stored.
Regarding customers, customers can book more than one car to a place. The customers are allowed to book multiple cars to different places, in a single booking transaction. The name,

address, no of passengers travelling in the car, the destination ,the rental cost etc needs to be stored.

The following constraints are to be defined for the vehicles, drivers, and destination places:

a) The vehicle make should be after the year 2000.
b) Only vehicles of maruti, Tata are used by the company
c) Drivers should be above 20 years of age
d) Drivers should be staying in "pune" city
e) The destination places should be within 500km radius from Pune.

Design the relational database for the above company, so that the following queries can be answered:

1. List the names of drivers who have driven a car to "Mumbai"
2. List the name of customers who have booked a "SUV" to "satara"
3. List the names of customers who have booked cars to pune or Mumbai or Lonavla
4. List the details of cars that have never driven down to "Mumbai"
5. List the details of the place to which maximum number of customers have driven down.
6. List the details of the driver who have driven all the vehicles of the company.
7. List the names of the drivers who have driven atleast two cars to "Mumbai
8. List the names of drivers who have also driven some vehicles to "Mumbai"
9. List the details of customers who have booked more than two vehicles to "solapur"
10. List the names of customers who have booked maximum number of vehicles

Signature of the instructor | **Date** | **Remark**