Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)

Academic Year: 2022-2023

Database Management System
EXPERIMENTS

Name:	Prerna Sunil Jadhav
Sap ID:	60004220127
Branch:	Computer Engineering
Semester:	III (DSE)
Course Code:	DJ19CEL304
Course:	Database Management Systems Laboratory
Experiments:	1 to 10

Name	Prerna Sunil Jadhav
SAP ID	60004220127
Experiment No.	01
Aim: Design an Entity Relational model	

Theory:
ER Model stands for Entity Relationship Model is a high-level conceptual data model diagram. ER model helps to systematically analyze data requirements to produce a well-designed database. The ER Model represents real-world entities and the relationships between them. Creating an ER Model in DBMS is considered as a best practice before implementing your database.

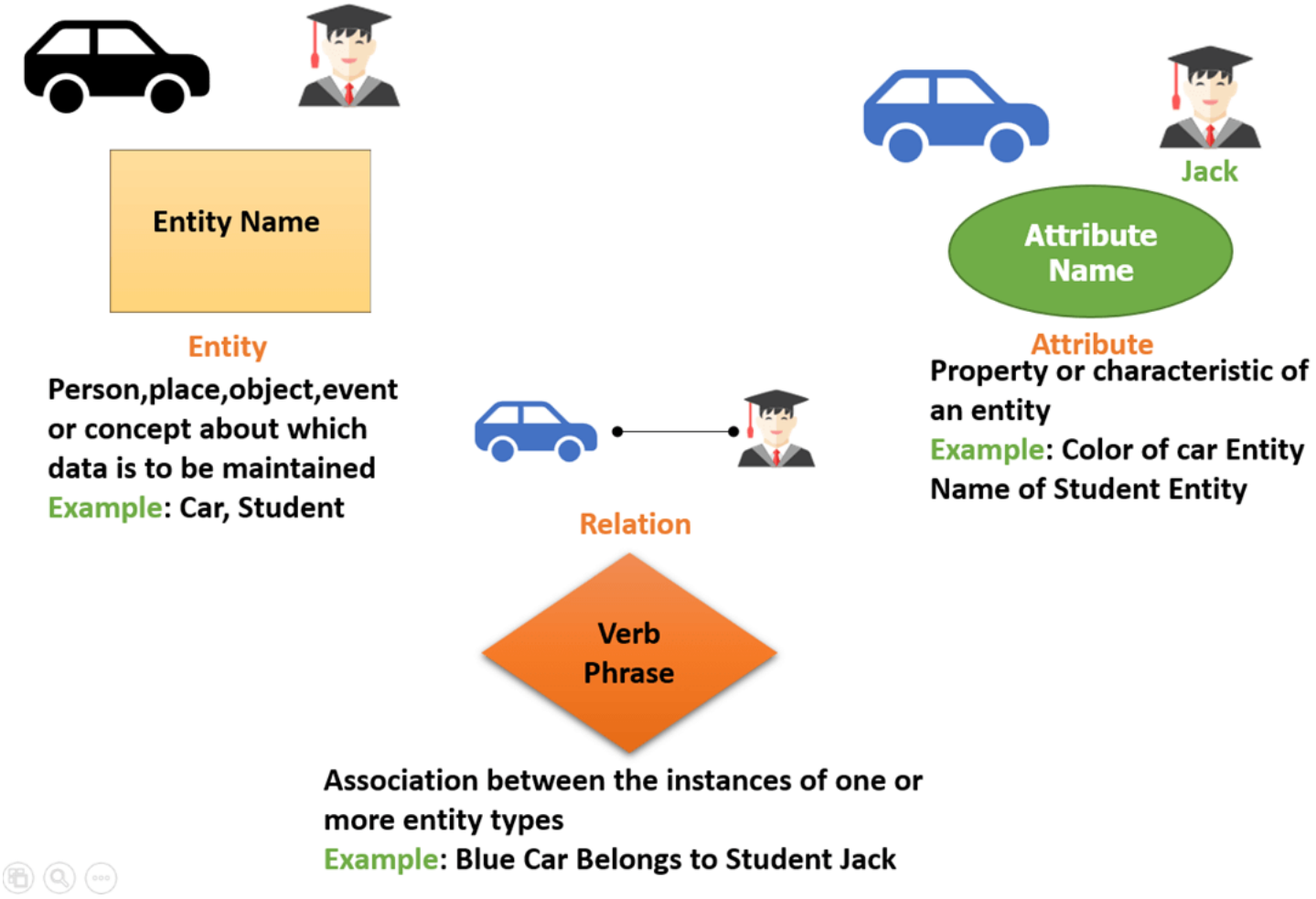
ER Modeling helps you to analyze data requirements systematically to produce a well-designed database. So, it is considered a best practice to complete ER modeling before implementing your database.

Here, are prime reasons for using the ER Diagram

- Helps you to define terms related to entity relationship modeling
- Provide a preview of how all your tables should connect, what fields are going to be on each table
- Helps to describe entities, attributes, relationships
- ER diagrams are translatable into relational tables which allows you to build databases quickly
- ER diagrams can be used by database designers as a blueprint for implementing data in specific software applications
- The database designer gains a better understanding of the information to be contained in the database with the help of ERP diagram
- ERD Diagram allows you to communicate with the logical structure of the database to users

Components of the ER Diagram
This model is based on three basic concepts:
Entities
Attributes
Relationships

For example, in a University database, we might have entities for Students, Courses, and Lecturers. Students entity can have attributes like Rollno, Name, and DeptID. They might have relationships with Courses and Lecturers.

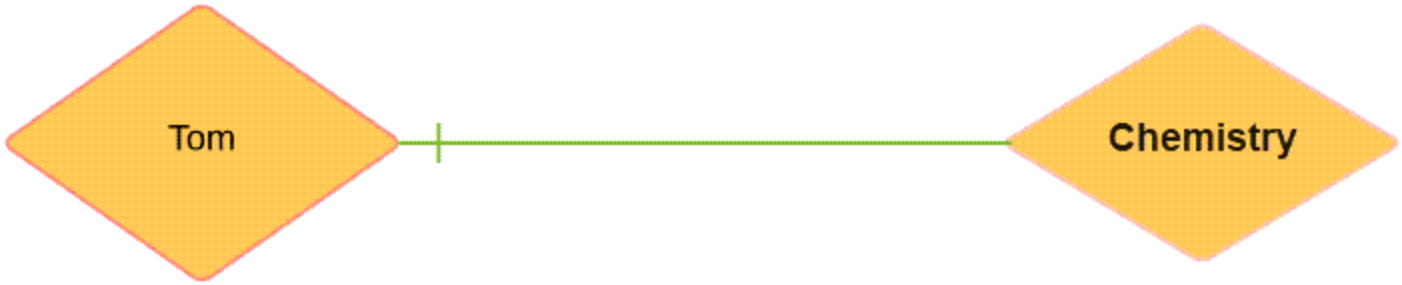


Entity set:
An entity set is a group of similar kind of entities. It may contain entities with attribute sharing similar values. Entities are represented by their properties, which also called attributes. All attributes have their separate values. For example, a student entity may have a name, age, class, as attributes.

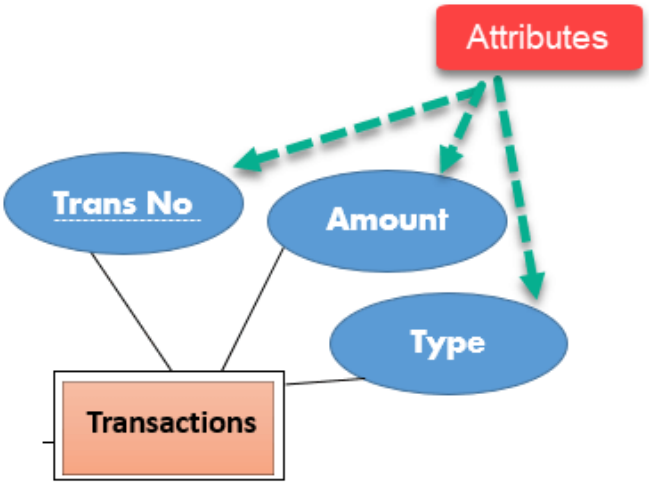
Example of Entities:
A university may have some departments. All these departments employ various lecturers and offer several programs.

Relationship
Relationship is nothing but an association among two or more entities. E.g., Tom works in the Chemistry department.
Entities take part in relationships. We can often identify relationships with verbs or verb phrases.

For example:
You are attending this lecture
I am giving the lecture
Just loke entities, we can classify relationships according to relationship-types:
A student attends a lecture
A lecturer is giving a lecture.



Attributes
It is a single-valued property of either an entity-type or a relationship-type.
For example, a lecture might have attributes: time, date, duration, place, etc.
An attribute in ER Diagram examples, is represented by an Ellipse



Cardinality

Defines the numerical attributes of the relationship between two entities or entity sets.

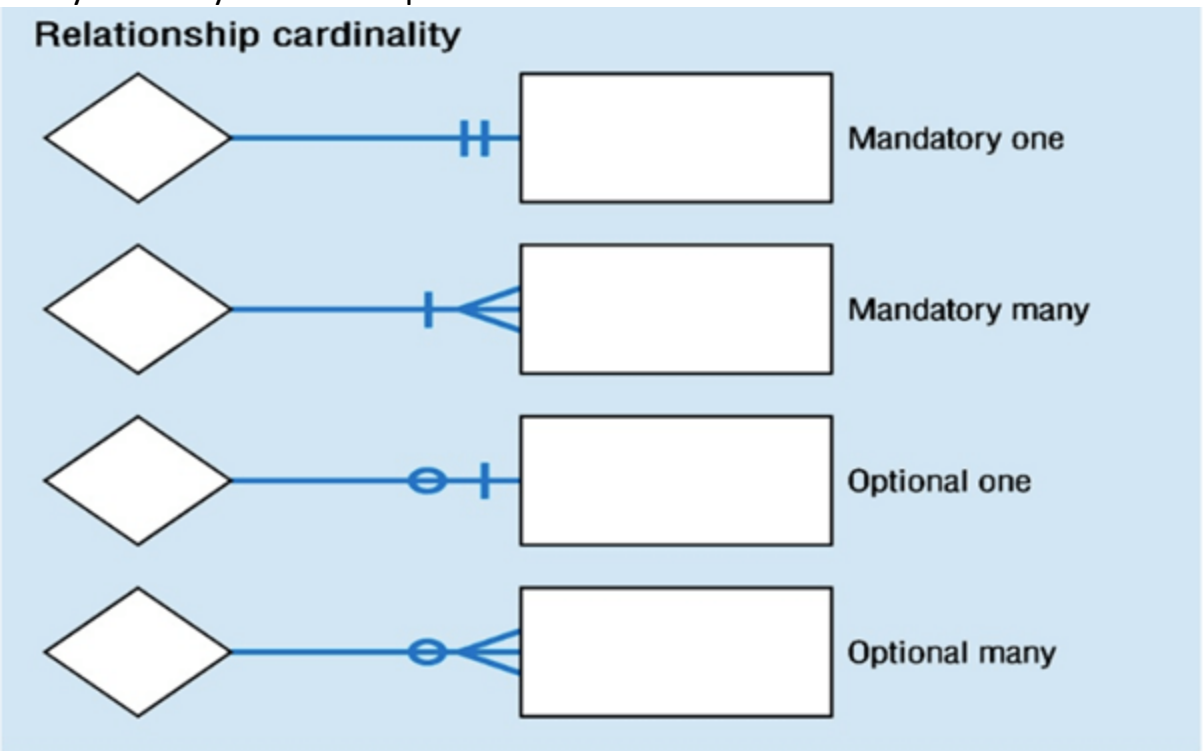
Different types of cardinal relationships are:

One-to-One Relationships

One-to-Many Relationships

May to One Relationships

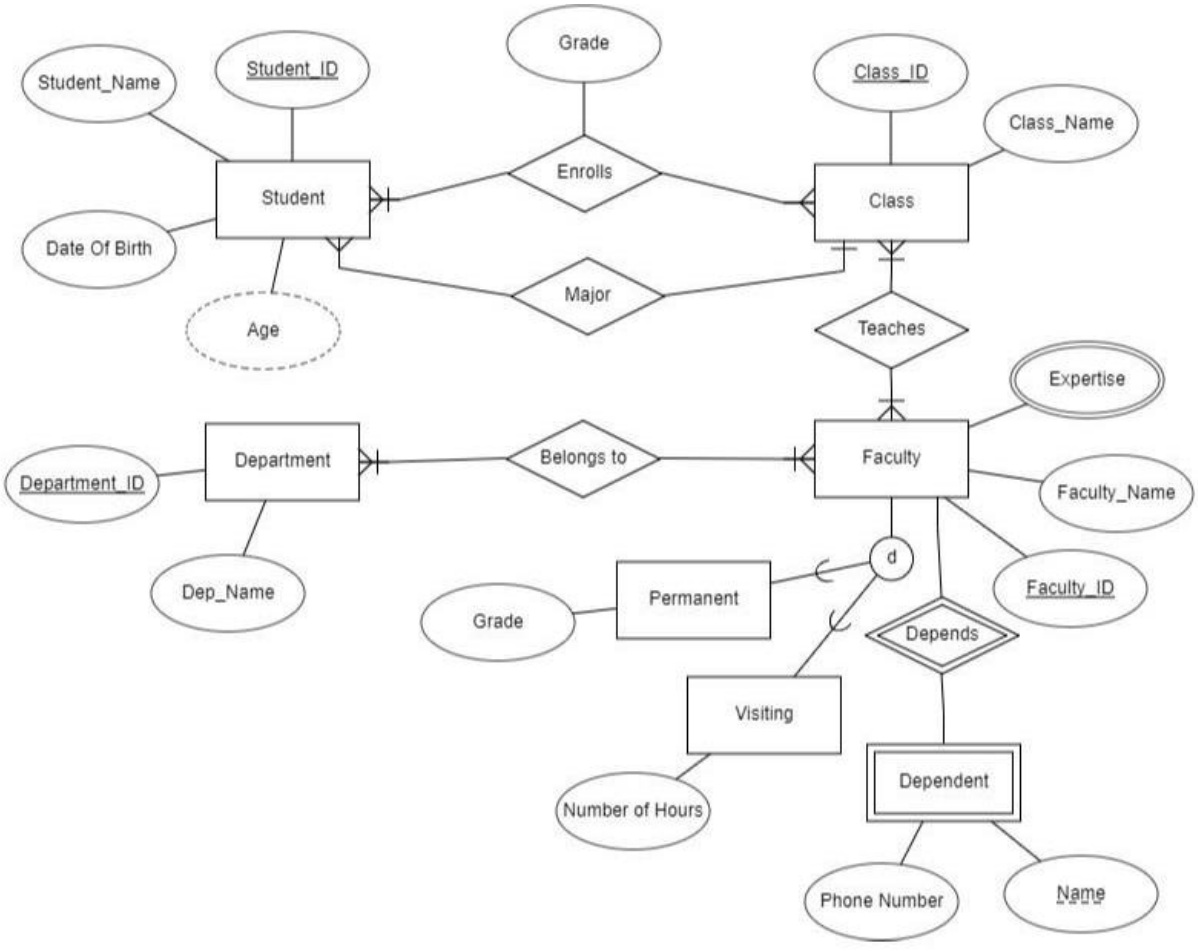
Many-to-Many Relationships



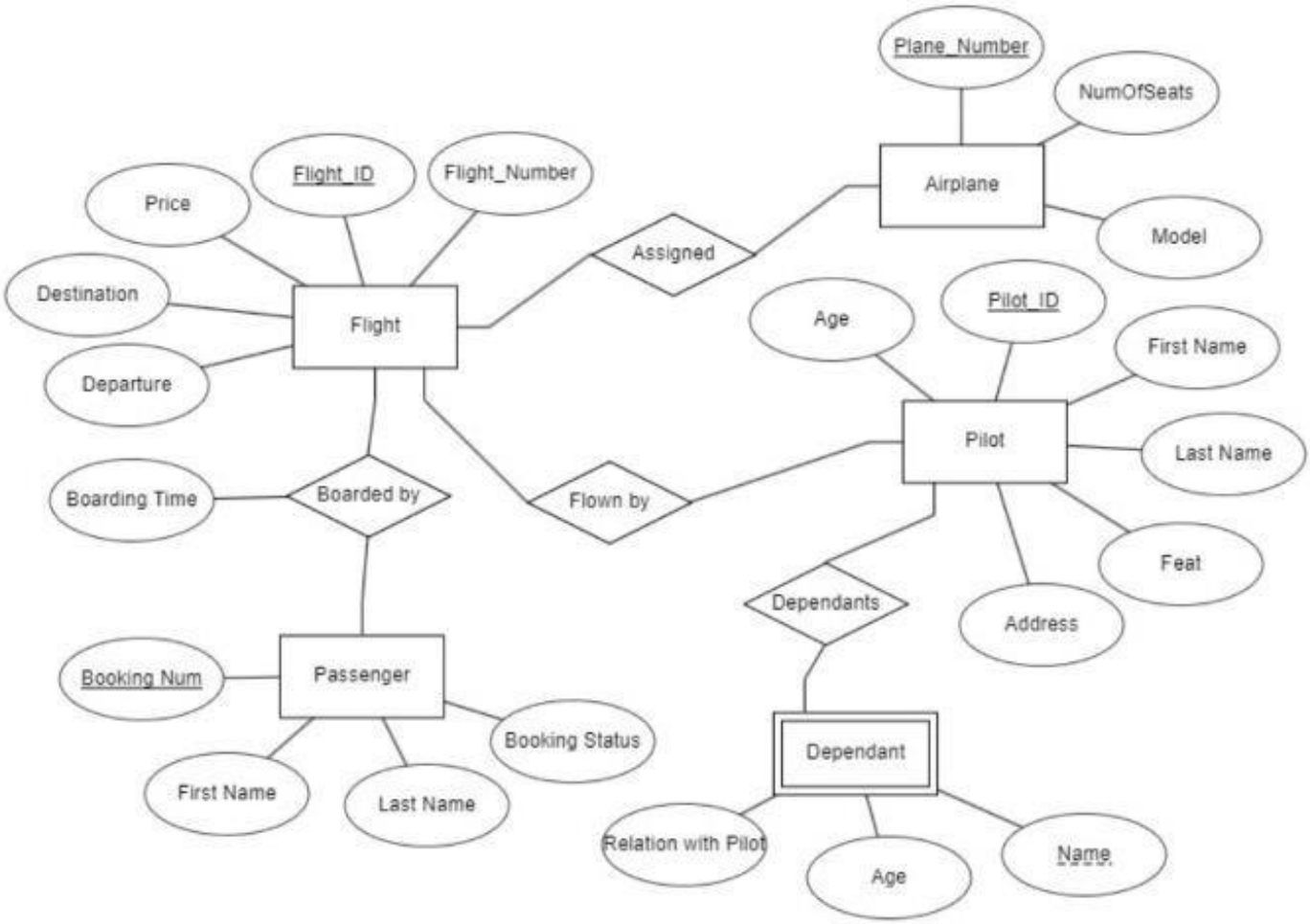
Following are the main components and its symbols in ER Diagrams:

- Rectangles: This Entity Relationship Diagram symbol represents entity types
- Ellipses : Symbol represent attributes
- Diamonds: This symbol represents relationship types
- Lines: It links attributes to entity types and entity types with other relationship types
- Primary key: attributes are underlined
- Double Ellipses: Represent multi-valued attributes

Example1 for ER:



Example2 for ER:



Conclusion: Thus, we designed an entity relationship model.

Name	Prerna Sunil Jadhav
SAP ID	60004220127
Experiment No.	02
Aim: Mapping ER/EER Relational Schema model	

Theory:

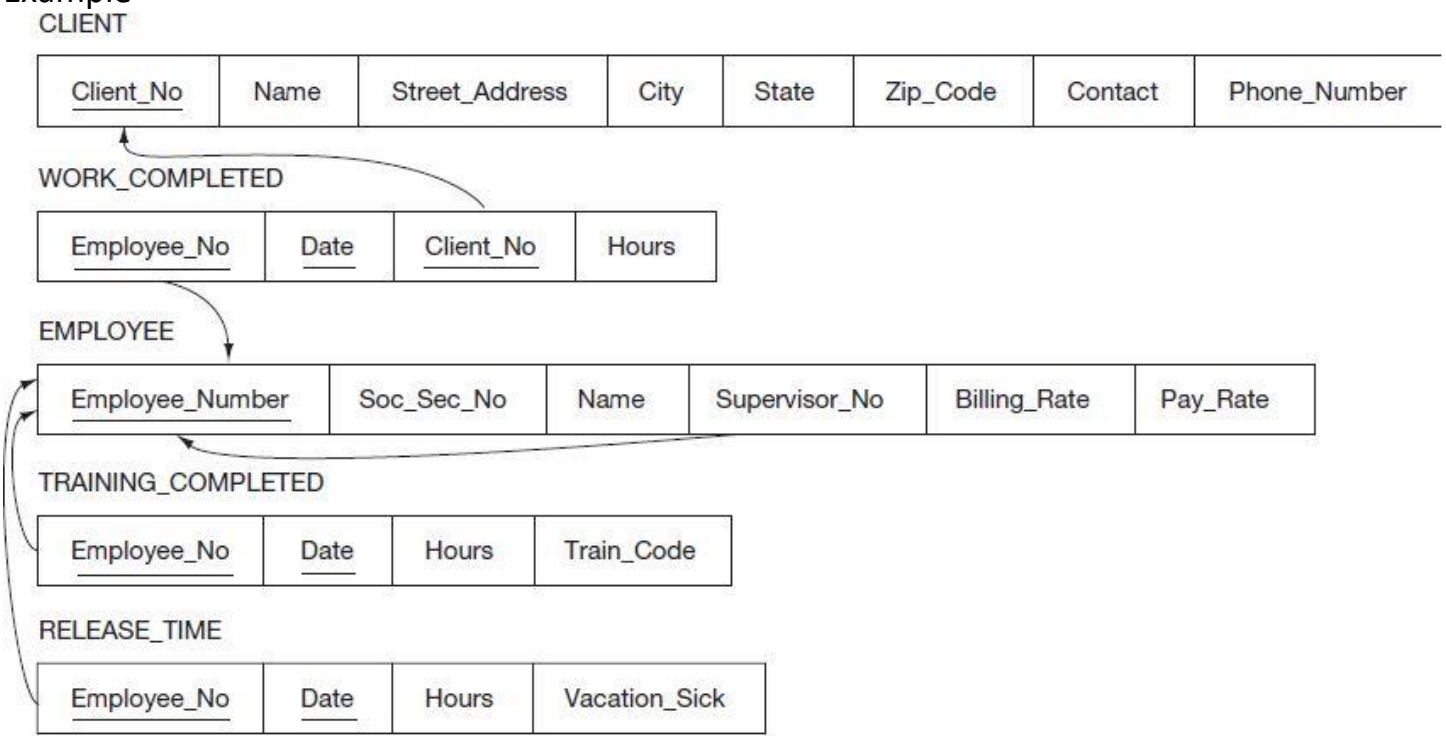
ER-to-Relational Mapping Algorithm

- Step 1: Mapping of Regular Entity Types
- Step 2: Mapping of Weak Entity Types
- Step 3: Mapping of Binary 1:1 Relation Types
- Step 4: Mapping of Binary 1:N Relationship Types.
- Step 5: Mapping of Binary M:N Relationship Types.
- Step 6: Mapping of Multivalued attributes.
- Step 7: Mapping of N-ary Relationship Types.

Mapping EER Model Constructs to Relations

- Step 8: Options for Mapping Specialization or Generalization.
- Step 9: Mapping of Union Types (Categories).

Example



Name	Prerna Sunil Jadhav
SAP ID	60004220127
Experiment No.	03
Aim: Create and populate database using Data Definition Language (DDL) and DML Commands for your specified System.	

Theory:

- **DDL-** Data Definition Language (DDL) statements are used to define the database structure or schema.
- Data Definition Language understanding with database schemas and describes how the data should consist in the database, therefore language statements like CREATE TABLE or ALTER TABLE belongs to the DDL. DDL is about "metadata".
- DDL includes commands such as CREATE, ALTER, and DROP statements. DDL is used to CREATE, ALTER, OR DROP the database objects (Table, Views, Users).
- Data Definition Language (DDL) are used different statements:
  1. CREATE - to create objects in the database
  2. ALTER - alters the structure of the database
  3. DROP - delete objects from the database
  4. TRUNCATE - remove all records from a table, including all spaces allocated for the records are removed
  5. RENAME - rename an object

Use Following schema to perform the experiment.
Student *(stuId, lastName, firstName, major, credits)*
Faculty *(facId, name, department, rank)*
Class *(classNumber, facId, schedule, room)*
Enroll *(classNumber, stuId, grade)*

NOTE: Underlined Text: Primary Key, Italic Text: Forign Key

DDL (Data Definition Language)

1. **CREATE**

   a. **To create a database**

   CREATE DATABASE *dbname*; Eg-
   CREATE DATABASE UniversityData;

   b. **To select existing database**

   Use dbname;
   Eg- USE UniversityData;

   c. **To create a Table**

   The create table command defines each column of the table uniquely. Each column hasminimum 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 statementis terminated with a semicolon.

   Syntax: Create table table name(fieldname1 datatype(),fieldname2 datatype()...);

2. **ALTER**

   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: Add a new column,cTitle, to our Classtable ALTER TABLE Class ADD cTitle CHAR(30);

The schema of the Class table would then be:
Class(classNumber,facId,schedule,room,cTitle)

Dropping a Column from the Table

Syntax:

ALTER TABLE <table_name> DROP COLUMN <column_name>

Example:

Example: Drop the cTitle column from the Class table ALTER TABLE Class DROP COLUMN cTitle;

This command will drop particular column

If we want to add, drop, or change a constraint, we can use the same ALTER TABLE command. For example, if we created the Class table and neglected to make facIda foreign key in Class, we could add the constraint at any time by writing:

ALTER TABLE Class ADD CONSTRAINT Class_facId_fk FOREIGN KEY (facId)REFERENCES Faculty (facId)ON DELETE NO ACTION);

We could drop an existing named constraint using the ALTER TABLE command. For example, to drop the check condition on the creditsattribute of Student that we created earlier, we could write:

ALTER TABLE Student DROP CONSTRAINT Student_credits_cc;

➢ **Modifying Existing Column**

Syntax:

ALTER TABLE <table_name> MODIFY <column_name> NewDataType>(<NewSize>)

Example:

ALTER TABLE Student MODIFY stuld Varchar(20);

➢ **Renaming Existing Table**

Syntax:

ALTER TABLE <table_name> RENAME <new_table_name>

Example:

ALTER TABLE student RENAME new_student;

ALTER TABLE new_student RENAME student; (To revert the change)

3. RENAME

Syntax:

RENAME TABLE <OldTableName> TO <NewTableName>

Example:

RENAME table visiting TO visiting_staff;

4. DROP

Syntax:

DROP TABLE <table_name>

Example:

DROP TABLE visiting_staff;

5. TRUNCATE

Syntax:

TRUNCATE TABLE <Table_name>

Example:

TRUNCATE TABLE visiting_staff;

6- SHOW

To check available databases and tables

Syntax

SHOW DATABASES;

SHOWTABLES;

7- DESCRIBE

To obtain information about table structure or query execution plans.

DESCRIBE < table_name>

DESC < table_name>

Example- DESC
Student;

- **Apply Integrity Constraints for the specified system**

MySQL CONSTRAINT is used to define rules to allow or restrict what values can be stored in columns. The purpose of inducing constraints is to enforce the integrity of a database.
MySQL CONSTRAINTS are used to limit the type of data that can be inserted into a table. MySQL CONSTRAINTS can be classified into two types - column level and table level.
The column level constraints can apply only to one column where as table level constraints are applied to the entire table.
MySQL CONSTRAINT is declared at the time of creating a table.

MySQL CONSTRAINTs are :

- NOT NULL
- UNIQUE
- PRIMARY KEY
- FOREIGN KEY
- CHECK
- DEFAULT
- AUTO INCREMENT

CONSTRAINT	DESCRIPTION
NOT NULL	In MySQL NOT NULL constraint allows to specify that a column can not contain any NULL value. MySQL NOT NULL can be used to CREATE and ALTER a table. Eg. lastName CHAR(20) NOT NULL, firstName CHAR(20) NOT NULL
UNIQUE	The UNIQUE constraint in MySQL does not allow to insert a duplicate value in a column. The UNIQUE constraint maintains the uniqueness of a column in a table. More than one UNIQUE column can be used in a table. Eg. CONSTRAINT Class_schedule_room_uk UNIQUE (schedule, room)
PRIMARY KEY	A PRIMARY KEY constraint for a table enforces the table to accept unique data for a specific column and this constraint creates a unique index for accessing the table faster. Eg: CONSTRAINT Faculty_facId_pk PRIMARY KEY (facId));
FOREIGN KEY	A FOREIGN KEY in MySQL creates a link between two tables by one specific column of both tables. The specified column in one table must be a PRIMARY KEY and referred by the column of another table knownas FOREIGN KEY. Eg. CONSTRAINT Class_facId_fk FOREIGN KEY (facId) REFERENCES Faculty (facId) ON DELETE NO ACTION);
CHECK	A CHECK constraint controls the values in the associated column. The CHECK constraint determines whether the value is valid or not from a logical expression. Eg: CONSTRAINT Student_credits_cc CHECK ((credits =0) AND (credits 150);
DEFAULT	In a MySQL table, each column must contain a value (including a NULL). While inserting data into a table, if no value is supplied to a column, then the column gets the value set as DEFAULT. credits SMALLINT DEFAULT 0 CHECK ((credits =0) AND (credits 150))
AUTO INCREMENT	Auto-increment allows a unique number to be generated automatically when a new record is inserted into a table. Often this is the primary key field that we would like to be created automatically every time a new record is inserted. Eg. Field name int AUTO_INCREMENT PRIMARY KEY

DML(Data Manipulation Language)

A data manipulation language (DML) is a family of computer languages including commands permitting users to manipulate data in a database. This manipulation involves inserting data into database tables, retrieving existing data, deleting data from existing tables and modifying existing data. DML is mostly incorporated in SQL databases.
Use following database to perform the experiment.

Database:

1) INSERT

This command adds one or more records to a database table.

Syntax

INSERT INTO "table_name" ("column1", "column2", ...) VALUES
("value1", "value2", ...);

Example

1) insert into Student (stuld, lastname, firstname, major, credits) values('S1001','Smith',Tom', 'History', 90);

2) SELECT

The SELECT statement is used to select data from a database.

Syntax

SELECT * FROM *table_name*;

Example-

Select * from Student;

3) UPDATE

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

Syntax

UPDATE *table_name*
SET *column1=value1*, *column2=value2*,...
WHERE *some_column=some_value*;

Example-

1) update Student set Name='JANEE' where stuID=S1020;

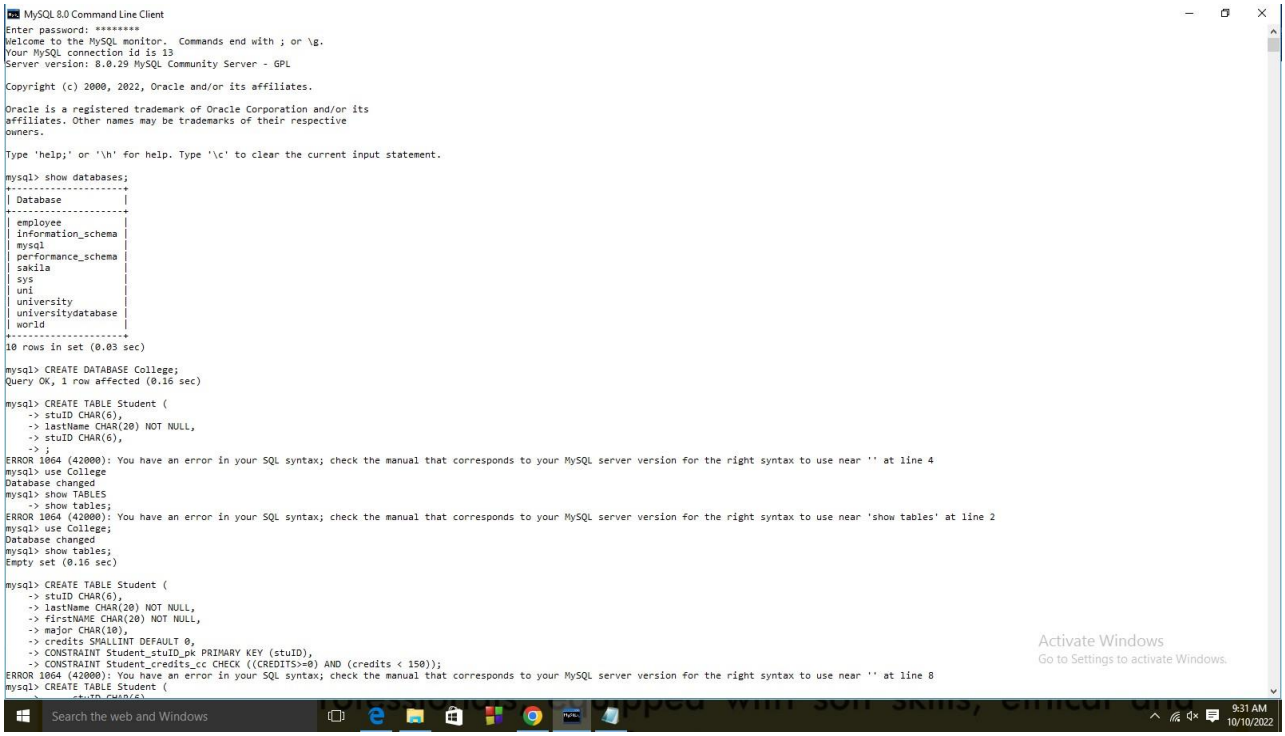
4)DELETE

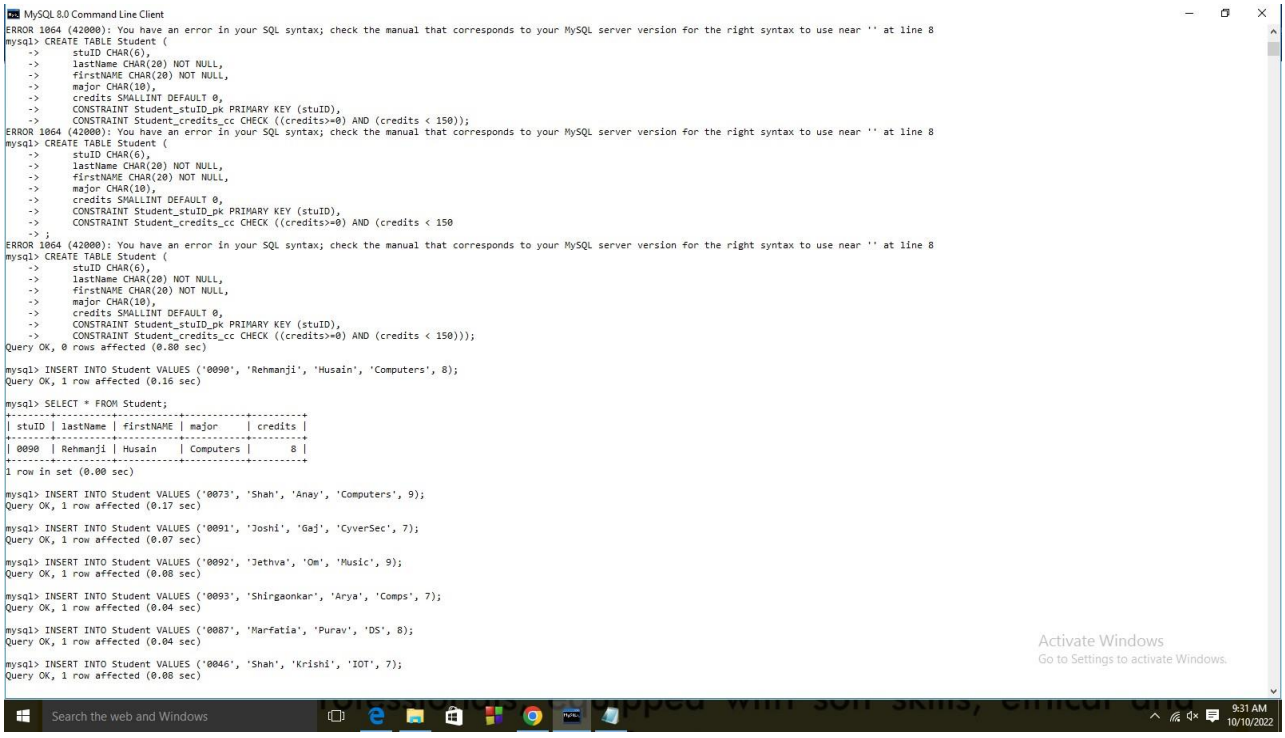
This command removes one or more records from a table according to specified conditions.

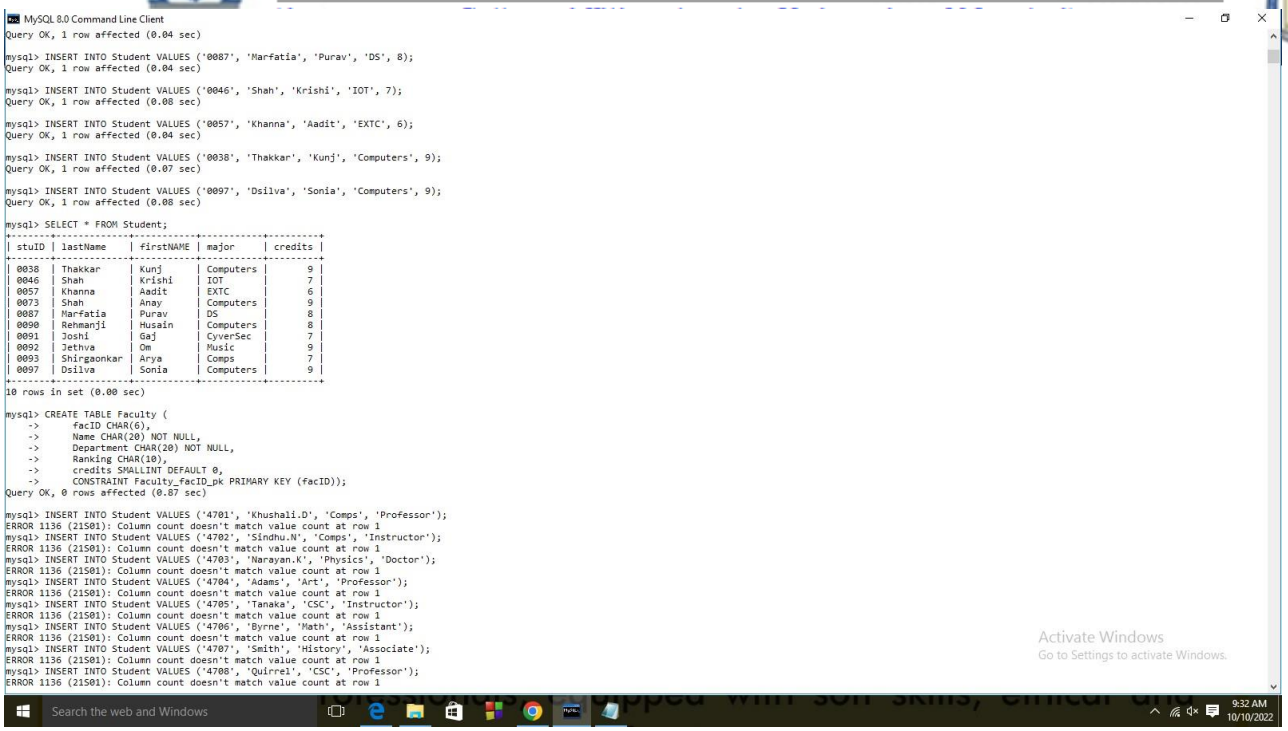
Syntax: DELETE FROM *table_name*
WHERE *some_column=some_value*;

Example-1) delete from Student
where stuID=S1020;

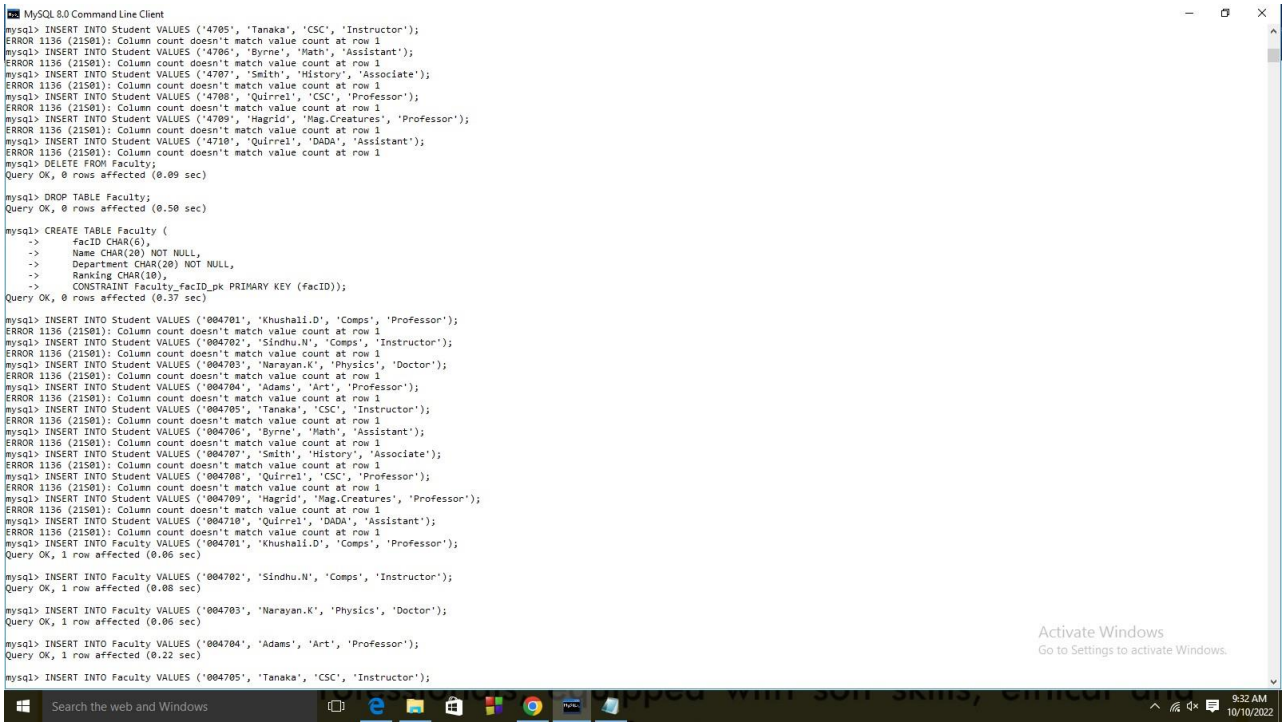
Conclusion: Data definition Language is used to create database and apply Integrity Constraints for the project and used DML to populate database.


MySQL 8.0 Command Line Client
Enter password: ********
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 13
Server version: 8.0.29 MySQL Community Server - GPL
Copyright (c) 2000, 2022, Oracle and/or its affiliates.
Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
mysql> show databases;
+--------------------+
| Database |
+--------------------+
| employee |
| information_schema |
| mysql |
| performance_schema |
| sakila |
| sys |
| uni |
| university |
| universitydatabase |
| world |
+--------------------+
10 rows in set (0.03 sec)
mysql> CREATE DATABASE College;
Query OK, 1 row affected (0.16 sec)
mysql> CREATE TABLE Student (
-> stuID CHAR(6),
-> lastName CHAR(20) NOT NULL,
-> stuID CHAR(6),
-> ;
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near '' at line 4
mysql> use College
Database changed
mysql> show TABLES
-> show tables;
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near 'show tables' at line 2
mysql> use College;
Database changed
mysql> show tables;
Empty set (0.16 sec)
mysql> CREATE TABLE Student (
-> stuID CHAR(6),
-> lastName CHAR(20) NOT NULL,
-> firstNAME CHAR(20) NOT NULL,
-> major CHAR(10),
-> credits SMALLINT DEFAULT 0,
-> CONSTRAINT Student_stuID_pk PRIMARY KEY (stuID),
-> CONSTRAINT Student_credits_cc CHECK ((CREDITS>=0) AND (credits < 150));
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near '' at line 8
mysql> CREATE TABLE Student (
Activate Windows
Go to Settings to activate Windows.
Search the web and Windows
9:31 AM
10/10/2022


MySQL 8.0 Command Line Client
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near '' at line 8
mysql> CREATE TABLE Student (
-> stuID CHAR(6),
-> lastName CHAR(20) NOT NULL,
-> firstNAME CHAR(20) NOT NULL,
-> major CHAR(10),
-> credits SMALLINT DEFAULT 0,
-> CONSTRAINT Student_stuID_pk PRIMARY KEY (stuID),
-> CONSTRAINT Student_credits_cc CHECK ((credits>=0) AND (credits < 150));
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near '' at line 8
mysql> CREATE TABLE Student (
-> stuID CHAR(6),
-> lastName CHAR(20) NOT NULL,
-> firstNAME CHAR(20) NOT NULL,
-> major CHAR(10),
-> credits SMALLINT DEFAULT 0,
-> CONSTRAINT Student_stuID_pk PRIMARY KEY (stuID),
-> CONSTRAINT Student_credits_cc CHECK ((credits>=0) AND (credits < 150
-> ;
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near '' at line 8
mysql> CREATE TABLE Student (
-> stuID CHAR(6),
-> lastName CHAR(20) NOT NULL,
-> firstNAME CHAR(20) NOT NULL,
-> major CHAR(10),
-> credits SMALLINT DEFAULT 0,
-> CONSTRAINT Student_stuID_pk PRIMARY KEY (stuID),
-> CONSTRAINT Student_credits_cc CHECK ((credits>=0) AND (credits < 150)));
Query OK, 0 rows affected (0.80 sec)
mysql> INSERT INTO Student VALUES ('0090', 'Rehmanji', 'Husain', 'Computers', 8);
Query OK, 1 row affected (0.16 sec)
mysql> SELECT * FROM Student;
+-------+----------+-----------+-----------+---------+
| stuID | lastName | firstNAME | major | credits |
+-------+----------+-----------+-----------+---------+
| 0090 | Rehmanji | Husain | Computers | 8 |
+-------+----------+-----------+-----------+---------+
1 row in set (0.00 sec)
mysql> INSERT INTO Student VALUES ('0073', 'Shah', 'Anay', 'Computers', 9);
Query OK, 1 row affected (0.17 sec)
mysql> INSERT INTO Student VALUES ('0091', 'Joshi', 'Gaj', 'CyverSec', 7);
Query OK, 1 row affected (0.07 sec)
mysql> INSERT INTO Student VALUES ('0092', 'Jethva', 'Om', 'Music', 9);
Query OK, 1 row affected (0.08 sec)
mysql> INSERT INTO Student VALUES ('0093', 'Shirgaonkar', 'Arya', 'Comps', 7);
Query OK, 1 row affected (0.04 sec)
mysql> INSERT INTO Student VALUES ('0087', 'Marfatia', 'Purav', 'DS', 8);
Query OK, 1 row affected (0.04 sec)
mysql> INSERT INTO Student VALUES ('0046', 'Shah', 'Krishi', 'IOT', 7);
Query OK, 1 row affected (0.08 sec)
Activate Windows
Go to Settings to activate Windows.
Search the web and Windows
9:31 AM
10/10/2022



```
MySQL 8.0 Command Line Client
Query OK, 1 row affected (0.04 sec)

mysql> INSERT INTO Student VALUES ('0087', 'Marfatia', 'Purav', 'DS', 8);
Query OK, 1 row affected (0.04 sec)

mysql> INSERT INTO Student VALUES ('0046', 'Shah', 'Krishi', 'IOT', 7);
Query OK, 1 row affected (0.08 sec)

mysql> INSERT INTO Student VALUES ('0057', 'Khanna', 'Aadit', 'EXTC', 6);
Query OK, 1 row affected (0.04 sec)

mysql> INSERT INTO Student VALUES ('0038', 'Thakkar', 'Kunj', 'Computers', 9);
Query OK, 1 row affected (0.07 sec)

mysql> INSERT INTO Student VALUES ('0097', 'Dsilva', 'Sonia', 'Computers', 9);
Query OK, 1 row affected (0.08 sec)

mysql> SELECT * FROM Student;
+-------+-------------+-----------+-----------+---------+
| stuID | lastName    | firstNAME | major     | credits |
+-------+-------------+-----------+-----------+---------+
| 0038  | Thakkar     | Kunj      | Computers |       9 |
| 0046  | Shah        | Krishi    | IOT       |       7 |
| 0057  | Khanna      | Aadit     | EXTC      |       6 |
| 0073  | Shah        | Anay      | Computers |       9 |
| 0087  | Marfatia    | Purav     | DS        |       8 |
| 0090  | Rehmanji    | Husain    | Computers |       8 |
| 0091  | Joshi       | Gaj       | CyverSec  |       7 |
| 0092  | Jethva      | Om        | Music     |       9 |
| 0093  | Shirgaonkar | Arya      | Comps     |       7 |
| 0097  | Dsilva      | Sonia     | Computers |       9 |
+-------+-------------+-----------+-----------+---------+
10 rows in set (0.00 sec)

mysql> CREATE TABLE Faculty (
    ->      facID CHAR(6),
    ->      Name CHAR(20) NOT NULL,
    ->      Department CHAR(20) NOT NULL,
    ->      Ranking CHAR(10),
    ->      credits SMALLINT DEFAULT 0,
    ->      CONSTRAINT Faculty_facID_pk PRIMARY KEY (facID));
Query OK, 0 rows affected (0.87 sec)

mysql> INSERT INTO Student VALUES ('4701', 'Khushali.D', 'Comps', 'Professor');
ERROR 1136 (21S01): Column count doesn't match value count at row 1
mysql> INSERT INTO Student VALUES ('4702', 'Sindhu.N', 'Comps', 'Instructor');
ERROR 1136 (21S01): Column count doesn't match value count at row 1
mysql> INSERT INTO Student VALUES ('4703', 'Narayan.K', 'Physics', 'Doctor');
ERROR 1136 (21S01): Column count doesn't match value count at row 1
mysql> INSERT INTO Student VALUES ('4704', 'Adams', 'Art', 'Professor');
ERROR 1136 (21S01): Column count doesn't match value count at row 1
mysql> INSERT INTO Student VALUES ('4705', 'Tanaka', 'CSC', 'Instructor');
ERROR 1136 (21S01): Column count doesn't match value count at row 1
mysql> INSERT INTO Student VALUES ('4706', 'Byrne', 'Math', 'Assistant');
ERROR 1136 (21S01): Column count doesn't match value count at row 1
mysql> INSERT INTO Student VALUES ('4707', 'Smith', 'History', 'Associate');
ERROR 1136 (21S01): Column count doesn't match value count at row 1
mysql> INSERT INTO Student VALUES ('4708', 'Quirrel', 'CSC', 'Professor');
ERROR 1136 (21S01): Column count doesn't match value count at row 1
```

```
MySQL 8.0 Command Line Client
mysql> INSERT INTO Student VALUES ('4705', 'Tanaka', 'CSC', 'Instructor');
ERROR 1136 (21S01): Column count doesn't match value count at row 1
mysql> INSERT INTO Student VALUES ('4706', 'Byrne', 'Math', 'Assistant');
ERROR 1136 (21S01): Column count doesn't match value count at row 1
mysql> INSERT INTO Student VALUES ('4707', 'Smith', 'History', 'Associate');
ERROR 1136 (21S01): Column count doesn't match value count at row 1
mysql> INSERT INTO Student VALUES ('4708', 'Quirrel', 'CSC', 'Professor');
ERROR 1136 (21S01): Column count doesn't match value count at row 1
mysql> INSERT INTO Student VALUES ('4709', 'Hagrid', 'Mag.Creatures', 'Professor');
ERROR 1136 (21S01): Column count doesn't match value count at row 1
mysql> INSERT INTO Student VALUES ('4710', 'Quirrel', 'DADA', 'Assistant');
ERROR 1136 (21S01): Column count doesn't match value count at row 1
mysql> DELETE FROM Faculty;
Query OK, 0 rows affected (0.09 sec)

mysql> DROP TABLE Faculty;
Query OK, 0 rows affected (0.50 sec)

mysql> CREATE TABLE Faculty (
    ->      facID CHAR(6),
    ->      Name CHAR(20) NOT NULL,
    ->      Department CHAR(20) NOT NULL,
    ->      Ranking CHAR(10),
    ->      CONSTRAINT Faculty_facID_pk PRIMARY KEY (facID));
Query OK, 0 rows affected (0.37 sec)

mysql> INSERT INTO Student VALUES ('004701', 'Khushali.D', 'Comps', 'Professor');
ERROR 1136 (21S01): Column count doesn't match value count at row 1
mysql> INSERT INTO Student VALUES ('004702', 'Sindhu.N', 'Comps', 'Instructor');
ERROR 1136 (21S01): Column count doesn't match value count at row 1
mysql> INSERT INTO Student VALUES ('004703', 'Narayan.K', 'Physics', 'Doctor');
ERROR 1136 (21S01): Column count doesn't match value count at row 1
mysql> INSERT INTO Student VALUES ('004704', 'Adams', 'Art', 'Professor');
ERROR 1136 (21S01): Column count doesn't match value count at row 1
mysql> INSERT INTO Student VALUES ('004705', 'Tanaka', 'CSC', 'Instructor');
ERROR 1136 (21S01): Column count doesn't match value count at row 1
mysql> INSERT INTO Student VALUES ('004706', 'Byrne', 'Math', 'Assistant');
ERROR 1136 (21S01): Column count doesn't match value count at row 1
mysql> INSERT INTO Student VALUES ('004707', 'Smith', 'History', 'Associate');
ERROR 1136 (21S01): Column count doesn't match value count at row 1
mysql> INSERT INTO Student VALUES ('004708', 'Quirrel', 'CSC', 'Professor');
ERROR 1136 (21S01): Column count doesn't match value count at row 1
mysql> INSERT INTO Student VALUES ('004709', 'Hagrid', 'Mag.Creatures', 'Professor');
ERROR 1136 (21S01): Column count doesn't match value count at row 1
mysql> INSERT INTO Student VALUES ('004710', 'Quirrel', 'DADA', 'Assistant');
ERROR 1136 (21S01): Column count doesn't match value count at row 1
mysql> INSERT INTO Faculty VALUES ('004701', 'Khushali.D', 'Comps', 'Professor');
Query OK, 1 row affected (0.06 sec)

mysql> INSERT INTO Faculty VALUES ('004702', 'Sindhu.N', 'Comps', 'Instructor');
Query OK, 1 row affected (0.08 sec)

mysql> INSERT INTO Faculty VALUES ('004703', 'Narayan.K', 'Physics', 'Doctor');
Query OK, 1 row affected (0.06 sec)

mysql> INSERT INTO Faculty VALUES ('004704', 'Adams', 'Art', 'Professor');
Query OK, 1 row affected (0.22 sec)

mysql> INSERT INTO Faculty VALUES ('004705', 'Tanaka', 'CSC', 'Instructor');
```

MySQL 8.0 Command Line Client

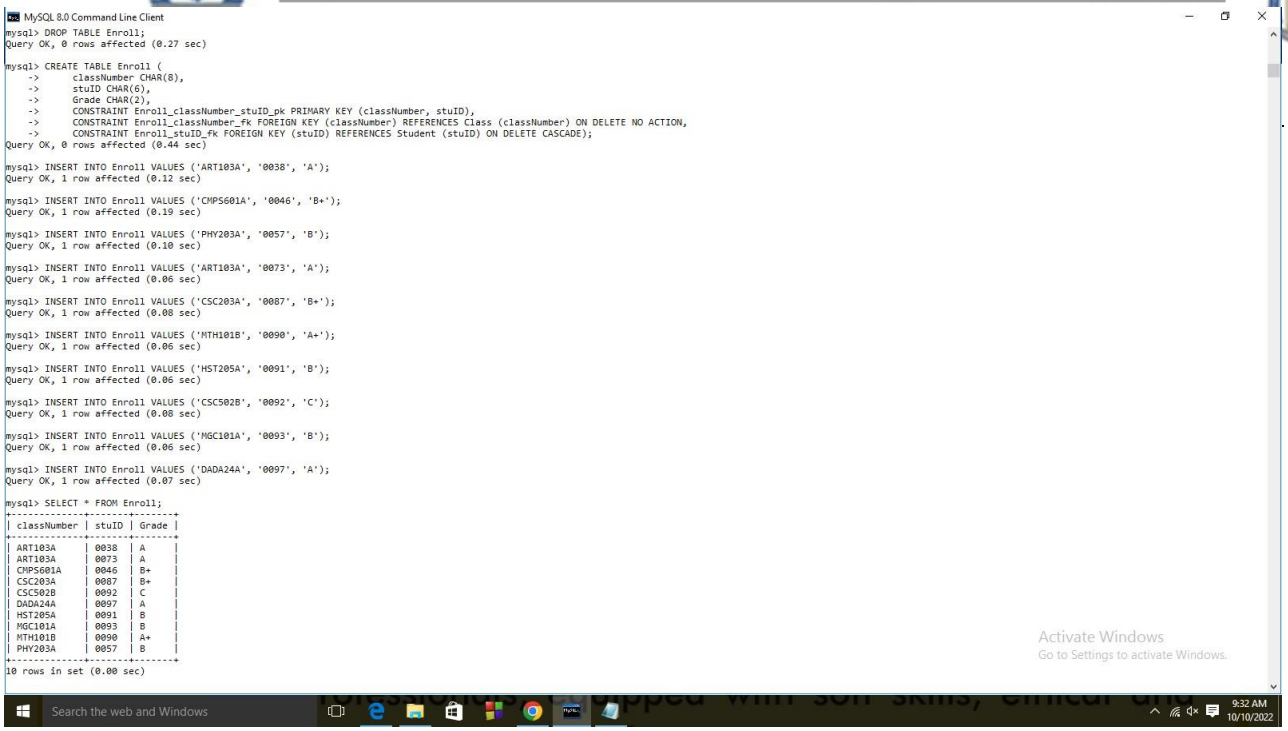
```
mysql> INSERT INTO Faculty VALUES ('004704', 'Adams', 'Art', 'Professor');
Query OK, 1 row affected (0.22 sec)

mysql> INSERT INTO Faculty VALUES ('004705', 'Tanaka', 'CSC', 'Instructor');
Query OK, 1 row affected (0.13 sec)

mysql> INSERT INTO Faculty VALUES ('004706', 'Byrne', 'Math', 'Assistant');
Query OK, 1 row affected (0.06 sec)

mysql> INSERT INTO Faculty VALUES ('004707', 'Smith', 'History', 'Associate');
Query OK, 1 row affected (0.07 sec)

mysql> INSERT INTO Faculty VALUES ('004708', 'Quirrel', 'CSC', 'Professor');
Query OK, 1 row affected (0.07 sec)

mysql> INSERT INTO Faculty VALUES ('004709', 'Hagrid', 'Mag.Creatures', 'Professor');
Query OK, 1 row affected (0.06 sec)

mysql> INSERT INTO Faculty VALUES ('004710', 'Quirrel', 'DADA', 'Assistant');
Query OK, 1 row affected (0.05 sec)

mysql> SELECT * FROM Faculty;
+--------+------------+---------------+------------+
| facID  | Name       | Department    | Ranking    |
+--------+------------+---------------+------------+
| 004701 | Khushali.D | Comps         | Professor  |
| 004702 | Sindhu.N   | Comps         | Instructor |
| 004703 | Narayan.K  | Physics       | Doctor     |
| 004704 | Adams      | Art           | Professor  |
| 004705 | Tanaka     | CSC           | Instructor |
| 004706 | Byrne      | Math          | Assistant  |
| 004707 | Smith      | History       | Associate  |
| 004708 | Quirrel    | CSC           | Professor  |
| 004709 | Hagrid     | Mag.Creatures | Professor  |
| 004710 | Quirrel    | DADA          | Assistant  |
+--------+------------+---------------+------------+
10 rows in set (0.00 sec)

mysql> CREATE TABLE Class (
    ->      classNumber CHAR(8),
    ->      facID CHAR(6) NOT NULL,
    ->      Schedule CHAR(8),
    ->      Room CHAR(6),
    ->      CONSTRAINT Class_classNumber_pk PRIMARY KEY (classNumber)
    ->      CONSTRAINT Class_facID_fk FOREIGN KEY (facID) REFERENCES Faculty (facID) ON DELETE NO ACTION);
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near 'CONSTRAINT Class_facID_fk FOREIGN KEY (facID) REFERENCES Faculty (facI
D) ON DELE' at line 7
mysql> CREATE TABLE Class (
    ->      classNumber CHAR(8),
    ->      facID CHAR(6) NOT NULL,
    ->      Schedule CHAR(8),
    ->      Room CHAR(6),
    ->      CONSTRAINT Class_classNumber_pk PRIMARY KEY (classNumber),
    ->      CONSTRAINT Class_facID_fk FOREIGN KEY (facID) REFERENCES Faculty (facID) ON DELETE NO ACTION);
Query OK, 0 rows affected (0.87 sec)

mysql> INSERT INTO Class VALUES ('CMPS103A', '004701', 'MWF9', 'H221');
Query OK, 1 row affected (0.15 sec)
```


Search the web and Windows 9:32 AM 10/10/2022

MySQL 8.0 Command Line Client

```
mysql> INSERT INTO Class VALUES ('CMPS103A', '004701', 'MWF9', 'H221');
Query OK, 1 row affected (0.15 sec)

mysql> INSERT INTO Class VALUES ('CMPS601A', '004702', 'TuThF10', 'M110');
Query OK, 1 row affected (0.07 sec)

mysql> INSERT INTO Class VALUES ('PHY203A', '004703', 'MThF12', 'M110');
Query OK, 1 row affected (0.07 sec)

mysql> INSERT INTO Class VALUES ('ART103A', '004704', 'MWF11', 'M112');
Query OK, 1 row affected (0.06 sec)

mysql> INSERT INTO Class VALUES ('CSC203A', '004705', 'MTuTh9', 'H225');
Query OK, 1 row affected (0.07 sec)

mysql> INSERT INTO Class VALUES ('MTH101B', '004706', 'WThSa8', 'H223');
Query OK, 1 row affected (0.06 sec)

mysql> INSERT INTO Class VALUES ('HST205A', '004707', 'MThF12', 'M110');
Query OK, 1 row affected (0.20 sec)

mysql> INSERT INTO Class VALUES ('CSC502B', '004708', 'MThF11', 'H223');
Query OK, 1 row affected (0.14 sec)

mysql> INSERT INTO Class VALUES ('MGC101A', '004709', 'MThF10', 'FBDFST');
Query OK, 1 row affected (0.08 sec)

mysql> INSERT INTO Class VALUES ('DADA24A', '004710', 'TuFSa9', 'D207');
Query OK, 1 row affected (0.11 sec)

mysql> SELECT * FROM Class;
+-------------+--------+----------+--------+
| classNumber | facID  | Schedule | Room   |
+-------------+--------+----------+--------+
| ART103A     | 004704 | MWF11    | M112   |
| CMPS103A    | 004701 | MWF9     | H221   |
| CMPS601A    | 004702 | TuThF10  | M110   |
| CSC203A     | 004705 | MTuTh9   | H225   |
| CSC502B     | 004708 | MThF11   | H223   |
| DADA24A     | 004710 | TuFSa9   | D207   |
| HST205A     | 004707 | MThF12   | M110   |
| MGC101A     | 004709 | MThF10   | FBDFST |
| MTH101B     | 004706 | WThSa8   | H223   |
| PHY203A     | 004703 | MThF12   | M110   |
+-------------+--------+----------+--------+
10 rows in set (0.06 sec)

mysql> CREATE TABLE Enroll (
    ->      classNumber CHAR(8),
    ->      facID CHAR(6) NOT NULL,
    ->      stuID CHAR(6),
    ->      Grade CHAR(2),
    ->      CONSTRAINT Enroll_classNumber_stuID_pk PRIMARY KEY (classNumber, stuID),
    ->      CONSTRAINT Enroll_classNumber_fk FOREIGN KEY (classNumber) REFERENCES Class (classNumber) ON DELETE NO ACTION,
    ->      CONSTRAINT Enroll_stuID_fk FOREIGN KEY (stuID) REFERENCES Student (stuID) ON DELETE CASCADE);
Query OK, 0 rows affected (0.56 sec)

mysql> DROP TABLE Enroll;
Query OK, 0 rows affected (0.27 sec)
```


Search the web and Windows 9:32 AM 10/10/2022


MySQL 8.0 Command Line Client
mysql> DROP TABLE Enroll;
Query OK, 0 rows affected (0.27 sec)
mysql> CREATE TABLE Enroll (
-> classNumber CHAR(8),
-> stuID CHAR(6),
-> Grade CHAR(2),
-> CONSTRAINT Enroll_classNumber_stuID_pk PRIMARY KEY (classNumber, stuID),
-> CONSTRAINT Enroll_classNumber_fk FOREIGN KEY (classNumber) REFERENCES Class (classNumber) ON DELETE NO ACTION,
-> CONSTRAINT Enroll_stuID_fk FOREIGN KEY (stuID) REFERENCES Student (stuID) ON DELETE CASCADE);
Query OK, 0 rows affected (0.44 sec)
mysql> INSERT INTO Enroll VALUES ('ART103A', '0038', 'A');
Query OK, 1 row affected (0.12 sec)
mysql> INSERT INTO Enroll VALUES ('CMPS601A', '0046', 'B+');
Query OK, 1 row affected (0.19 sec)
mysql> INSERT INTO Enroll VALUES ('PHY203A', '0057', 'B');
Query OK, 1 row affected (0.10 sec)
mysql> INSERT INTO Enroll VALUES ('ART103A', '0073', 'A');
Query OK, 1 row affected (0.06 sec)
mysql> INSERT INTO Enroll VALUES ('CSC203A', '0087', 'B+');
Query OK, 1 row affected (0.08 sec)
mysql> INSERT INTO Enroll VALUES ('MTH101B', '0090', 'A+');
Query OK, 1 row affected (0.06 sec)
mysql> INSERT INTO Enroll VALUES ('HST205A', '0091', 'B');
Query OK, 1 row affected (0.06 sec)
mysql> INSERT INTO Enroll VALUES ('CSC502B', '0092', 'C');
Query OK, 1 row affected (0.08 sec)
mysql> INSERT INTO Enroll VALUES ('MGC101A', '0093', 'B');
Query OK, 1 row affected (0.06 sec)
mysql> INSERT INTO Enroll VALUES ('DADA24A', '0097', 'A');
Query OK, 1 row affected (0.07 sec)
mysql> SELECT * FROM Enroll;
classNumber	stuID	Grade
ART103A	0038	A
ART103A	0073	A
CMPS601A	0046	B+
CSC203A	0087	B+
CSC502B	0092	C
DADA24A	0097	A
HST205A	0091	B
MGC101A	0093	B
MTH101B	0090	A+
PHY203A	0057	B
10 rows in set (0.00 sec)
Activate Windows
Go to Settings to activate Windows.
Search the web and Windows
9:32 AM
10/10/2022

Name	Prerna Sunil Jadhav
SAP ID	60004220127
Experiment No.	04
Aim: To perform simple queries and string manipulation operations.	

THEORY:

A query is a request for data or information from a database table or combination of tables. This data may be generated as results returned by Structured Query Language (SQL) or as pictorials, graphs or complex results, e.g., trend analyses from data- mining tools.

For a machine to understand a request for information in the first place, the query must be written according to a code known as query language. SQL represents one of the standard languages used for database management purposes, while MySQL, instead, is the software using that specific language. Although SQL is a fairly universal query language, other commonly used ones include DMX, Datalog and AQL.

The most commonly used SQL command is SELECT statement. SQL SELECT statement is used to query or retrieve data from a table in the database. A query may retrieve information from specified columns or from all of the columns in the table. To create a simple SQL SELECT Statement, you must specify the column(s) name and the table name. The whole query is called SQL SELECT Statement.

Syntax-

SELECT [DISTINCT|ALL] { * | [fieldExpression [AS newName]} FROM tableName [alias] [WHERE condition][GROUP BY fieldName(s)] [HAVING condition] ORDER BY fieldName(s) .

- **SELECT** is the SQL keyword that lets the database know that you want to retrieve data.
- **[DISTINCT | ALL]** are optional keywords that can be used to fine tune the results returned from the SQL SELECT statement. If nothing is specified then ALL is assumed as the default.
- **{*| [fieldExpression [AS newName]}** at least one part must be specified, "*" selected all the fields from the specified table name, fieldExpression performs some computations on the specified fields such as adding numbers or putting together two string fields into one.
- **FROM** tableName is mandatory and must contain at least one table, multiple tables must be separated using commas or joined using the JOIN keyword.
- **WHERE** condition is optional, it can be used to specify criteria in the result set returned from the query.

- **GROUP BY** is used to put together records that have the same field values.

- **HAVING** condition is used to specify criteria when working using the GROUP BY keyword.
- **ORDER BY** is used to specify the sort order of the result set.

Queries:

1. Select all column from a table
   Select * from tablename;
2. Select specific column of list from a table
   Select col1, col2 from tablename;
3. Select …. where clause with various operators (<,>,<=,> =,IN,NOT IN,BETWEEN..AND,NOT BETWEEN… AND)
   Select col1 from tablename where col2>value1;
   Select * from tablename where col2 in(value1,value2,value3) Select * from tablename where col2 between value1 and value2
   Select * from tablename where col2 NOT between value1 and value2

4. Select ….where clause with multiple conditions
   Select * from tablename where col2=Value1 and/or col3=value2

5. Select …. where clause with string matching
   - Starts with any character
   - ends with any character
   - have a specific substring
   - character at specific position
   - starts with a specific character and having specifically n no of characters
   - starts and ends with different character

6. Query your database by applying aggregate function
   MIN, MAX, COUNT, AVG, SUM (create alias also)

7. Query your database by applying group by clause using one column and multiple column.

8. Query your database by applying order by clause using one column and multiple column

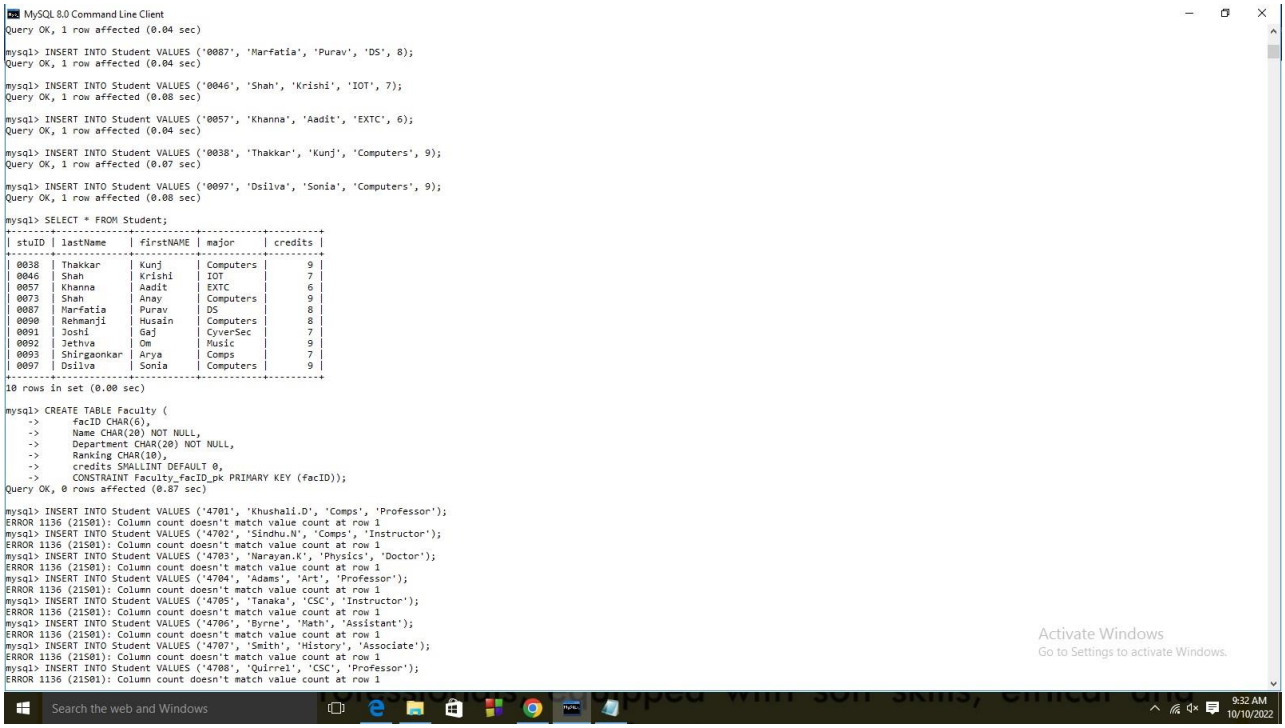
9. Query your database by applying group by, order by and having clause simultaneously

10. Select clause with various date functions:

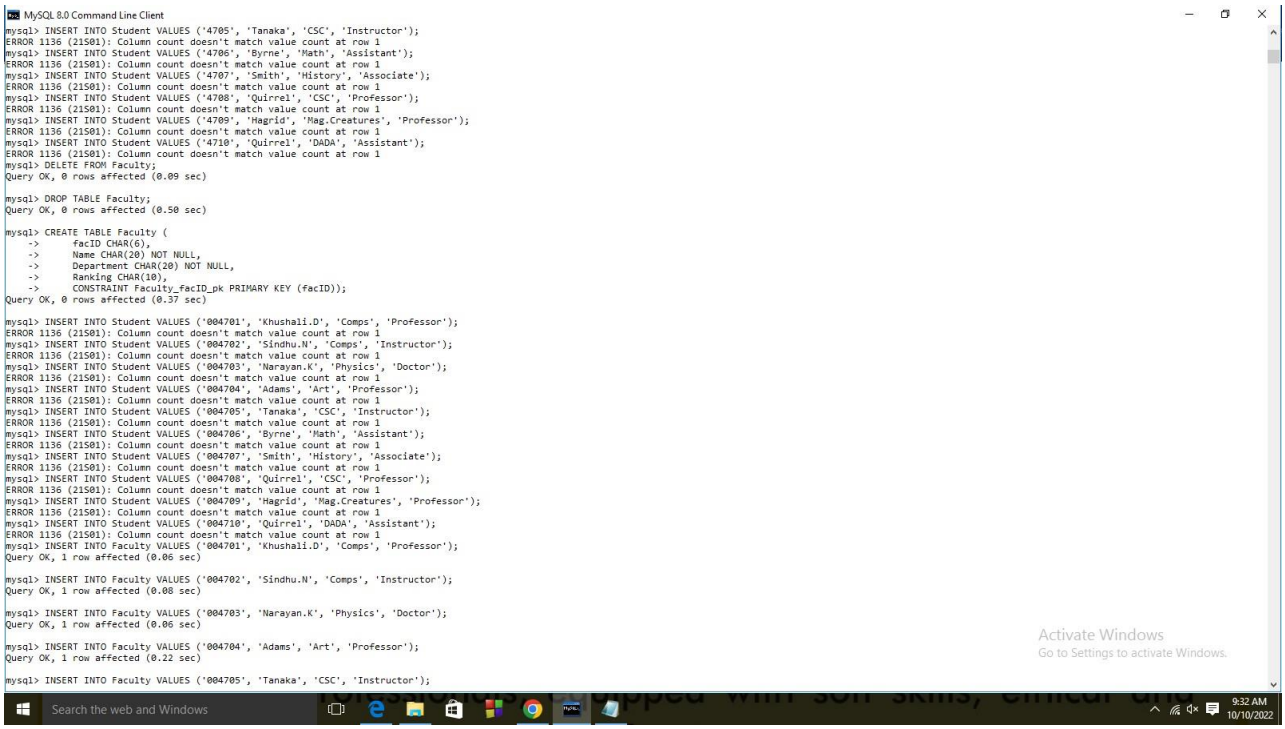
CURDATE(),CURRENT_TIME(),CURRENT_TIMESTAMP(),DATE(),DATEDIFF(),DAY(),DAYNAME(),EXTRACT(),MINUTE(),MONTH(),WEEK(),NOW(),YEAR()

11. Perform set operation on multiple select queries(UNION)

Screenshots on MYSQL:



```
MySQL 8.0 Command Line Client
Query OK, 1 row affected (0.04 sec)

mysql> INSERT INTO Student VALUES ('0087', 'Narfatia', 'Purav', 'DS', 8);
Query OK, 1 row affected (0.04 sec)

mysql> INSERT INTO Student VALUES ('0046', 'Shah', 'Krishi', 'IOT', 7);
Query OK, 1 row affected (0.08 sec)

mysql> INSERT INTO Student VALUES ('0057', 'Khanna', 'Aadit', 'EXTC', 6);
Query OK, 1 row affected (0.04 sec)

mysql> INSERT INTO Student VALUES ('0038', 'Thakkar', 'Kunj', 'Computers', 9);
Query OK, 1 row affected (0.07 sec)

mysql> INSERT INTO Student VALUES ('0097', 'Dsilva', 'Sonia', 'Computers', 9);
Query OK, 1 row affected (0.08 sec)

mysql> SELECT * FROM Student;
+-------+------------+-----------+-----------+---------+
| stuID | lastName   | firstNAME | major     | credits |
+-------+------------+-----------+-----------+---------+
| 0038  | Thakkar    | Kunj      | Computers |       9 |
| 0046  | Shah       | Krishi    | IOT       |       7 |
| 0057  | Khanna     | Aadit     | EXTC      |       6 |
| 0073  | Shah       | Anay      | Computers |       9 |
| 0087  | Narfatia   | Purav     | DS        |       8 |
| 0090  | Rehmanji   | Husain    | Computers |       8 |
| 0091  | Joshi      | Gaj       | CyverSec  |       7 |
| 0092  | Jethva     | Om        | Music     |       9 |
| 0093  | Shirgaonkar| Arya      | Comps     |       7 |
| 0097  | Dsilva     | Sonia     | Computers |       9 |
+-------+------------+-----------+-----------+---------+
10 rows in set (0.00 sec)

mysql> CREATE TABLE Faculty (
    ->      facID CHAR(6),
    ->      Name CHAR(20) NOT NULL,
    ->      Department CHAR(20) NOT NULL,
    ->      Ranking CHAR(10),
    ->      credits SMALLINT DEFAULT 0,
    ->      CONSTRAINT Faculty_facID_pk PRIMARY KEY (facID));
Query OK, 0 rows affected (0.87 sec)

mysql> INSERT INTO Student VALUES ('4701', 'Khushali.D', 'Comps', 'Professor');
ERROR 1136 (21S01): Column count doesn't match value count at row 1
mysql> INSERT INTO Student VALUES ('4702', 'Sindhu.N', 'Comps', 'Instructor');
ERROR 1136 (21S01): Column count doesn't match value count at row 1
mysql> INSERT INTO Student VALUES ('4703', 'Narayan.K', 'Physics', 'Doctor');
ERROR 1136 (21S01): Column count doesn't match value count at row 1
mysql> INSERT INTO Student VALUES ('4704', 'Adams', 'Art', 'Professor');
ERROR 1136 (21S01): Column count doesn't match value count at row 1
mysql> INSERT INTO Student VALUES ('4705', 'Tanaka', 'CSC', 'Instructor');
ERROR 1136 (21S01): Column count doesn't match value count at row 1
mysql> INSERT INTO Student VALUES ('4706', 'Byrne', 'Math', 'Assistant');
ERROR 1136 (21S01): Column count doesn't match value count at row 1
mysql> INSERT INTO Student VALUES ('4707', 'Smith', 'History', 'Associate');
ERROR 1136 (21S01): Column count doesn't match value count at row 1
mysql> INSERT INTO Student VALUES ('4708', 'Quirrel', 'CSC', 'Professor');
ERROR 1136 (21S01): Column count doesn't match value count at row 1
```

```
MySQL 8.0 Command Line Client
mysql> INSERT INTO Student VALUES ('4705', 'Tanaka', 'CSC', 'Instructor');
ERROR 1136 (21S01): Column count doesn't match value count at row 1
mysql> INSERT INTO Student VALUES ('4706', 'Byrne', 'Math', 'Assistant');
ERROR 1136 (21S01): Column count doesn't match value count at row 1
mysql> INSERT INTO Student VALUES ('4707', 'Smith', 'History', 'Associate');
ERROR 1136 (21S01): Column count doesn't match value count at row 1
mysql> INSERT INTO Student VALUES ('4708', 'Quirrel', 'CSC', 'Professor');
ERROR 1136 (21S01): Column count doesn't match value count at row 1
mysql> INSERT INTO Student VALUES ('4709', 'Hagrid', 'Mag.Creatures', 'Professor');
ERROR 1136 (21S01): Column count doesn't match value count at row 1
mysql> INSERT INTO Student VALUES ('4710', 'Quirrel', 'DADA', 'Assistant');
ERROR 1136 (21S01): Column count doesn't match value count at row 1
mysql> DELETE FROM Faculty;
Query OK, 0 rows affected (0.09 sec)

mysql> DROP TABLE Faculty;
Query OK, 0 rows affected (0.50 sec)

mysql> CREATE TABLE Faculty (
    ->      facID CHAR(6),
    ->      Name CHAR(20) NOT NULL,
    ->      Department CHAR(20) NOT NULL,
    ->      Ranking CHAR(10),
    ->      CONSTRAINT Faculty_facID_pk PRIMARY KEY (facID));
Query OK, 0 rows affected (0.37 sec)

mysql> INSERT INTO Student VALUES ('004701', 'Khushali.D', 'Comps', 'Professor');
ERROR 1136 (21S01): Column count doesn't match value count at row 1
mysql> INSERT INTO Student VALUES ('004702', 'Sindhu.N', 'Comps', 'Instructor');
ERROR 1136 (21S01): Column count doesn't match value count at row 1
mysql> INSERT INTO Student VALUES ('004703', 'Narayan.K', 'Physics', 'Doctor');
ERROR 1136 (21S01): Column count doesn't match value count at row 1
mysql> INSERT INTO Student VALUES ('004704', 'Adams', 'Art', 'Professor');
ERROR 1136 (21S01): Column count doesn't match value count at row 1
mysql> INSERT INTO Student VALUES ('004705', 'Tanaka', 'CSC', 'Instructor');
ERROR 1136 (21S01): Column count doesn't match value count at row 1
mysql> INSERT INTO Student VALUES ('004706', 'Byrne', 'Math', 'Assistant');
ERROR 1136 (21S01): Column count doesn't match value count at row 1
mysql> INSERT INTO Student VALUES ('004707', 'Smith', 'History', 'Associate');
ERROR 1136 (21S01): Column count doesn't match value count at row 1
mysql> INSERT INTO Student VALUES ('004708', 'Quirrel', 'CSC', 'Professor');
ERROR 1136 (21S01): Column count doesn't match value count at row 1
mysql> INSERT INTO Student VALUES ('004709', 'Hagrid', 'Mag.Creatures', 'Professor');
ERROR 1136 (21S01): Column count doesn't match value count at row 1
mysql> INSERT INTO Student VALUES ('004710', 'Quirrel', 'DADA', 'Assistant');
ERROR 1136 (21S01): Column count doesn't match value count at row 1
mysql> INSERT INTO Faculty VALUES ('004701', 'Khushali.D', 'Comps', 'Professor');
Query OK, 1 row affected (0.06 sec)

mysql> INSERT INTO Faculty VALUES ('004702', 'Sindhu.N', 'Comps', 'Instructor');
Query OK, 1 row affected (0.08 sec)

mysql> INSERT INTO Faculty VALUES ('004703', 'Narayan.K', 'Physics', 'Doctor');
Query OK, 1 row affected (0.06 sec)

mysql> INSERT INTO Faculty VALUES ('004704', 'Adams', 'Art', 'Professor');
Query OK, 1 row affected (0.22 sec)

mysql> INSERT INTO Faculty VALUES ('004705', 'Tanaka', 'CSC', 'Instructor');
```

MySQL 8.0 Command Line Client

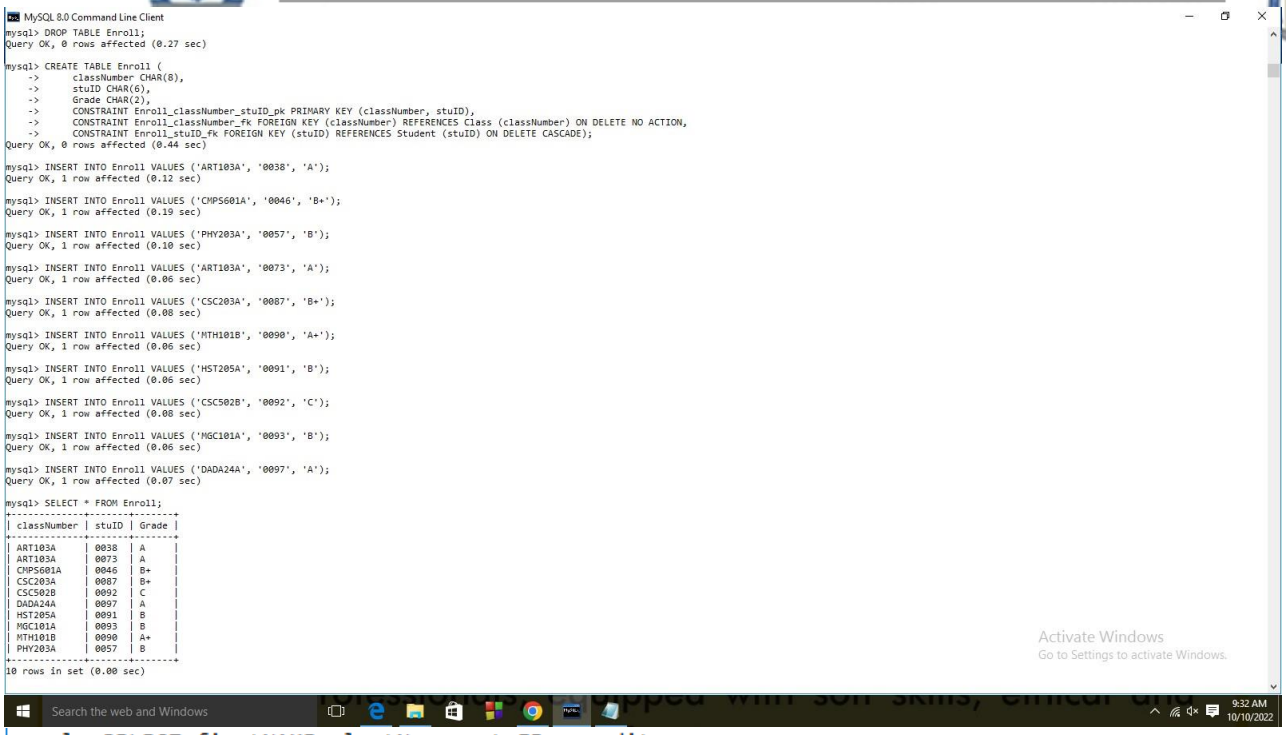
```
mysql> INSERT INTO Faculty VALUES ('004704', 'Adams', 'Art', 'Professor');
Query OK, 1 row affected (0.22 sec)

mysql> INSERT INTO Faculty VALUES ('004705', 'Tanaka', 'CSC', 'Instructor');
Query OK, 1 row affected (0.13 sec)

mysql> INSERT INTO Faculty VALUES ('004706', 'Byrne', 'Math', 'Assistant');
Query OK, 1 row affected (0.06 sec)

mysql> INSERT INTO Faculty VALUES ('004707', 'Smith', 'History', 'Associate');
Query OK, 1 row affected (0.07 sec)

mysql> INSERT INTO Faculty VALUES ('004708', 'Quirrel', 'CSC', 'Professor');
Query OK, 1 row affected (0.07 sec)

mysql> INSERT INTO Faculty VALUES ('004709', 'Hagrid', 'Mag.Creatures', 'Professor');
Query OK, 1 row affected (0.06 sec)

mysql> INSERT INTO Faculty VALUES ('004710', 'Quirrel', 'DADA', 'Assistant');
Query OK, 1 row affected (0.05 sec)

mysql> SELECT * FROM Faculty;
+--------+------------+---------------+------------+
| facID  | Name       | Department    | Ranking    |
+--------+------------+---------------+------------+
| 004701 | Khushali.D | Comps         | Professor  |
| 004702 | Sindhu.N   | Comps         | Instructor |
| 004703 | Narayan.K  | Physics       | Doctor     |
| 004704 | Adams      | Art           | Professor  |
| 004705 | Tanaka     | CSC           | Instructor |
| 004706 | Byrne      | Math          | Assistant  |
| 004707 | Smith      | History       | Associate  |
| 004708 | Quirrel    | CSC           | Professor  |
| 004709 | Hagrid     | Mag.Creatures | Professor  |
| 004710 | Quirrel    | DADA          | Assistant  |
+--------+------------+---------------+------------+
10 rows in set (0.00 sec)

mysql> CREATE TABLE Class (
    ->      classNumber CHAR(8),
    ->      facID CHAR(6) NOT NULL,
    ->      Schedule CHAR(8),
    ->      Room CHAR(6),
    ->      CONSTRAINT Class_classNumber_pk PRIMARY KEY (classNumber)
    ->      CONSTRAINT Class_facID_fk FOREIGN KEY (facID) REFERENCES Faculty (facID) ON DELETE NO ACTION);
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near 'CONSTRAINT Class_facID_fk FOREIGN KEY (facID) REFERENCES Faculty (facI
D) ON DELE' at line 7
mysql> CREATE TABLE Class (
    ->      classNumber CHAR(8),
    ->      facID CHAR(6) NOT NULL,
    ->      Schedule CHAR(8),
    ->      Room CHAR(6),
    ->      CONSTRAINT Class_classNumber_pk PRIMARY KEY (classNumber),
    ->      CONSTRAINT Class_facID_fk FOREIGN KEY (facID) REFERENCES Faculty (facID) ON DELETE NO ACTION);
Query OK, 0 rows affected (0.87 sec)

mysql> INSERT INTO Class VALUES ('CMPS103A', '004701', 'MWF9', 'H221');
Query OK, 1 row affected (0.15 sec)
```


Search the web and Windows 9:32 AM 10/10/2022

MySQL 8.0 Command Line Client

```
mysql> INSERT INTO Class VALUES ('CMPS103A', '004701', 'MWF9', 'H221');
Query OK, 1 row affected (0.15 sec)

mysql> INSERT INTO Class VALUES ('CMPS601A', '004702', 'TuThF10', 'M110');
Query OK, 1 row affected (0.07 sec)

mysql> INSERT INTO Class VALUES ('PHY203A', '004703', 'MThF12', 'M110');
Query OK, 1 row affected (0.07 sec)

mysql> INSERT INTO Class VALUES ('ART103A', '004704', 'MWF11', 'M112');
Query OK, 1 row affected (0.06 sec)

mysql> INSERT INTO Class VALUES ('CSC203A', '004705', 'MTuTh9', 'H225');
Query OK, 1 row affected (0.07 sec)

mysql> INSERT INTO Class VALUES ('MTH101B', '004706', 'WThSa8', 'H223');
Query OK, 1 row affected (0.06 sec)

mysql> INSERT INTO Class VALUES ('HST205A', '004707', 'MThF12', 'M110');
Query OK, 1 row affected (0.20 sec)

mysql> INSERT INTO Class VALUES ('CSC502B', '004708', 'MThF11', 'H223');
Query OK, 1 row affected (0.14 sec)

mysql> INSERT INTO Class VALUES ('MGC101A', '004709', 'MThF10', 'FBDFST');
Query OK, 1 row affected (0.08 sec)

mysql> INSERT INTO Class VALUES ('DADA24A', '004710', 'TuFSa9', 'D207');
Query OK, 1 row affected (0.11 sec)

mysql> SELECT * FROM Class;
+-------------+--------+----------+--------+
| classNumber | facID  | Schedule | Room   |
+-------------+--------+----------+--------+
| ART103A     | 004704 | MWF11    | M112   |
| CMPS103A    | 004701 | MWF9     | H221   |
| CMPS601A    | 004702 | TuThF10  | M110   |
| CSC203A     | 004705 | MTuTh9   | H225   |
| CSC502B     | 004708 | MThF11   | H223   |
| DADA24A     | 004710 | TuFSa9   | D207   |
| HST205A     | 004707 | MThF12   | M110   |
| MGC101A     | 004709 | MThF10   | FBDFST |
| MTH101B     | 004706 | WThSa8   | H223   |
| PHY203A     | 004703 | MThF12   | M110   |
+-------------+--------+----------+--------+
10 rows in set (0.06 sec)

mysql> CREATE TABLE Enroll (
    ->      classNumber CHAR(8),
    ->      facID CHAR(6) NOT NULL,
    ->      stuID CHAR(6),
    ->      Grade CHAR(2),
    ->      CONSTRAINT Enroll_classNumber_stuID_pk PRIMARY KEY (classNumber, stuID),
    ->      CONSTRAINT Enroll_classNumber_fk FOREIGN KEY (classNumber) REFERENCES Class (classNumber) ON DELETE NO ACTION,
    ->      CONSTRAINT Enroll_stuID_fk FOREIGN KEY (stuID) REFERENCES Student (stuID) ON DELETE CASCADE);
Query OK, 0 rows affected (0.56 sec)

mysql> DROP TABLE Enroll;
Query OK, 0 rows affected (0.27 sec)
```


Search the web and Windows 9:32 AM 10/10/2022




```
mysql> SELECT firstNAME, lastName, stuID, credits
    -> FROM Student
    -> WHERE major='Computers';
+-----------+----------+-------+---------+
| firstNAME | lastName | stuID | credits |
+-----------+----------+-------+---------+
| Kunj      | Thakkar  | 0038  |       9 |
| Anay      | Shah     | 0073  |       9 |
| Husain    | Rehmanji | 0090  |       8 |
| Sonia     | Dsilva   | 0097  |       9 |
+-----------+----------+-------+---------+
4 rows in set (0.00 sec)

mysql> SELECT * FROM Faculty WHERE Department='CSC';
+--------+---------+------------+------------+
| facID  | Name    | Department | Ranking    |
+--------+---------+------------+------------+
| 004705 | Tanaka  | CSC        | Instructor |
| 004708 | Quirrel | CSC        | Professor  |
+--------+---------+------------+------------+
2 rows in set (0.00 sec)

mysql> SELECT DISTINCT classNumber FROM Enroll;
+-------------+
| classNumber |
+-------------+
| ART103A     |
| CMPS601A    |
| CSC203A     |
| CSC502B     |
| DADA24A     |
| HST205A     |
| MGC101A     |
| MTH101B     |
| PHY203A     |
+-------------+
9 rows in set (0.31 sec)
```

```
mysql> SELECT * FROM Student;
+-------+-------------+-----------+-----------+---------+
| stuID | lastName    | firstNAME | major     | credits |
+-------+-------------+-----------+-----------+---------+
| 0038  | Thakkar     | Kunj      | Computers |       9 |
| 0046  | Shah        | Krishi    | IOT       |       7 |
| 0057  | Khanna      | Aadit     | EXTC      |       6 |
| 0073  | Shah        | Anay      | Computers |       9 |
| 0087  | Marfatia    | Purav     | DS        |       8 |
| 0090  | Rehmanji    | Husain    | Computers |       8 |
| 0091  | Joshi       | Gaj       | CyverSec  |       7 |
| 0092  | Jethva      | Om        | Music     |       9 |
| 0093  | Shirgaonkar | Arya      | Comps     |       7 |
| 0097  | Dsilva      | Sonia     | Computers |       9 |
+-------+-------------+-----------+-----------+---------+
10 rows in set (0.00 sec)

mysql> SELECT Name AS FacultyName, facID AS FacultyNumber
    -> FROM Faculty;
+-------------+---------------+
| FacultyName | FacultyNumber |
+-------------+---------------+
| Khushali.D  | 004701        |
| Sindhu.N    | 004702        |
| Narayan.K   | 004703        |
| Adams       | 004704        |
| Tanaka      | 004705        |
| Byrne       | 004706        |
| Smith       | 004707        |
| Quirrel     | 004708        |
| Hagrid      | 004709        |
| Quirrel     | 004710        |
+-------------+---------------+
10 rows in set (0.00 sec)

mysql> SELECT Name AS FacultyName, facID AS FacultyNumber
    -> FROM Faculty
    -> ORDER BY Name;
+-------------+---------------+
| FacultyName | FacultyNumber |
+-------------+---------------+
| Adams       | 004704        |
| Byrne       | 004706        |
| Hagrid      | 004709        |
| Khushali.D  | 004701        |
| Narayan.K   | 004703        |
| Quirrel     | 004708        |
| Quirrel     | 004710        |
| Sindhu.N    | 004702        |
| Smith       | 004707        |
| Tanaka      | 004705        |
```

```
mysql> SELECT firstNAME, lastName
    -> FROM Student
    -> WHERE major='Computers' AND credits>8;
+-----------+----------+
| firstNAME | lastName |
+-----------+----------+
| Kunj      | Thakkar  |
| Anay      | Shah     |
| Sonia     | Dsilva   |
+-----------+----------+
3 rows in set (0.00 sec)

mysql> SELECT Enroll.stuID, firstNAME, lastName
    -> FROm Enroll, Student
    -> WHERE classNumber='ART103A' AND Enroll.stuID=Student.stuID;
+-------+-----------+----------+
| stuID | firstNAME | lastName |
+-------+-----------+----------+
| 0038  | Kunj      | Thakkar  |
| 0073  | Anay      | Shah     |
+-------+-----------+----------+
2 rows in set (0.00 sec)

mysql> SELECT Enroll.stuID, firstNAME, lastName
    -> FROm Enroll, Student
    -> WHERE classNumber='ART103A' AND Student.stuID=Enroll.stuID;
+-------+-----------+----------+
| stuID | firstNAME | lastName |
+-------+-----------+----------+
| 0038  | Kunj      | Thakkar  |
| 0073  | Anay      | Shah     |
+-------+-----------+----------+
2 rows in set (0.00 sec)

mysql> SELECT stuID, Grade
    -> FROM Enroll, Class
    -> WHERE facID='004704' AND Class.classNumber=Enroll.classNumber
    -> ORDER BY stuID;
+-------+-------+
| stuID | Grade |
+-------+-------+
| 0038  | A     |
| 0073  | A     |
+-------+-------+
2 rows in set (0.06 sec)
```

```
mysql> SELECT Enroll.classNumber, firstNAME, lastName, major
    -> FROM Class, Enroll, Student
    -> WHERE facID='004704' AND Class.classNumber=Enroll.classNumber AND Enroll.stuID=Student.stuID;
+-------------+-----------+----------+-----------+
| classNumber | firstNAME | lastName | major     |
+-------------+-----------+----------+-----------+
| ART103A     | Kunj      | Thakkar  | Computers |
| ART103A     | Anay      | Shah     | Computers |
+-------------+-----------+----------+-----------+
2 rows in set (0.00 sec)

mysql> SELECT COPY1.classNumber, COPY1.Schedule, COPY1.Room, COPY2.classNumber, COPY2.Schedule
    -> FROM Class COPY1, Class COPY2
    -> WHERE COPY1.Room=COPY2.Room AND COPY1.classNumber>COPY2.classNumber;
+-------------+----------+------+-------------+----------+
| classNumber | Schedule | Room | classNumber | Schedule |
+-------------+----------+------+-------------+----------+
| HST205A     | MThF12   | M110 | CMPS601A    | TuThF10  |
| MTH101B     | WThSa8   | H223 | CSC502B     | MThF11   |
| PHY203A     | MThF12   | M110 | CMPS601A    | TuThF10  |
| PHY203A     | MThF12   | M110 | HST205A     | MThF12   |
+-------------+----------+------+-------------+----------+
4 rows in set (0.00 sec)
```

Name	Prerna Sunil Jadhav
SAP ID	60004220127
Experiment No.	05
Aim: Perform Nested queries and Complex queries.	

THEORY:

A Subquery or Inner query or Nested query is a query within another SQL query and embedded within the WHERE clause.

A subquery is used to return data that will be used in the main query as a condition to further restrict the data to be retrieved.

Subqueries can be used with the SELECT, INSERT, UPDATE, and DELETE statements along with the operators like =, , >=, <=, IN, BETWEEN etc.

SELECT column_name [, column_name] FROM table1
[, table2]
WHERE column_name OPERATOR (SELECT
column_name [, column_name] FROM table1 [, table2
]
[WHERE])

Example: Using a Subquery with Equality

Question:Find the numbers of all the courses taught by Byrne of the math department.

Solution:We already know how to do this by using a natural join, but there is another way of finding the solution. Instead of imagining a join from which we choose records with the same facId, we could visualize this as two separate queries. For the first one, we would go to the Faculty table and find the record with name of Byrne and department of Math. We could make a note of the corresponding facId. Then we could take the result of that query, namely Fll0, and search the Class table for records with that value in facId. Once we found them, we would display the classNumber. SQL allows us to sequence these queries so that the result of the first can be used in the second.

A subquery can be used in place of a join, provided the result to be displayed is contained in a single table and the data retrieved from the subquery con-sists of only one column. When you write a subquery involving two tables, you name only one table in each SELECT. The query to be done first, the subquery, is the one in parentheses, following the first WHERE line. The main query is performed using the result of the subquery. Normally you want the value of some field in the table mentioned in the main query to match the value of some field from the table in the subquery. In this example, we knew we would get only one value from the subquery, since facId is the key of Faculty, so a unique value would be produced. Therefore, we were able to use equality as the operator.
Since the sub-query is performed first, the SELECT . . . FROM . . . WHERE of the subquery is actually replaced by the value retrieved.

Example: Subquery Using 'IN'

Question:Find the names and IDs of all Faculty members who teach a class in Room H221.

Solution:We need two tables, Class and Faculty, to answer this question. We also see that the names and IDs both appear on the Faculty table, so we have a choice of a join or a subquery. If we use a subquery, we begin with the Class table to find facId values for any courses that meet in Room H221. We find two such entries, so we make a note of those values. Then we go to the Facultytable and compare the facId value of each record on that table with the two ID values from Class, and display the corresponding facId and name.
In the WHERE line of the main query we used IN, rather than =, because the result of the subquery is a set of values rather than a single value. We are saying we want the facId in Faculty to match any member of the set of values we obtain from the subquery.
The IN is a more general form of subquery than the comparison operator, which is restricted to the case where a single value is produced. We can also use the

negative form 'NOT IN', which will evaluate to true if the record has a field value which is not in the set of values retrieved by the subquery.

Example: Nested Subqueries

Question: Get an alphabetical list of names and IDs of all students in any class taught by F110.

Solution: We need three tables, Student, Enroll, and Class, to answer this question. However, the values to be displayed appear on one table, Student, so we can use a subquery. First we check the Class table to find the classNumber of all courses taught byF110. We find two values, MTH101B and MTH103C. Next we go to the Enroll table to find the stuId of all students in either of these courses. We find three values, S1020, S1010, and S1002. We nowlook at the Studenttable to find the records with matching stuId values, and display the stuId, lastName, and firstName, in alphabetical order by name.

```
mysql> SELECT classNumber
    -> FROM Class
    -> WHERE facID=(SELECT facID FROM Faculty WHERE Name='Byrne' AND Department='Math');
+--------------+
| classNumber |
+--------------+
| MTH101B     |
+--------------+
1 row in set (0.07 sec)
```

```
mysql> SELECT Name, facID
    -> FROM Faculty WHERE facID IN (SELECT facID FROM Class WHERE Room='H221');
+-------------+--------+
| Name        | facID  |
+-------------+--------+
| Khushali.D | 004701 |
+-------------+--------+
1 row in set (0.32 sec)
```

```
mysql> SELECT firstNAME, lastName, stuID
    -> FROM Student
    -> WHERE stuID IN (SELECT stuID FROM Enroll WHERE classNumber IN (SELECT classNumber FROM Class WHERE facID='004709'))
    -> ORDER BY firstNAME, lastName;
+-----------+-------------+-------+
| firstNAME | lastName    | stuID |
+-----------+-------------+-------+
| Arya      | Shirgaonkar | 0093  |
+-----------+-------------+-------+
```

Name	Prerna Sunil Jadhav
SAP ID	60004220127
Experiment No.	06
Aim: Perform Join operations.	

THEORY:

1) UNION

The UNION operator is used to combine the result-set of two or more SELECT statements.
Notice that each SELECT statement within the UNION must have the same number of
columns. The columns must also have similar data types. Also, the columns in each SELECT
statement must be in the same order.
SQL UNION Syntax

```
SELECT column_name(s) FROM table1
UNION
SELECT column_name(s) FROM table2;
```

Note: The UNION operator selects only distinct values by default.
To allow duplicate values,
use the ALL keyword with UNION.
SQL UNION ALL Syntax

```
SELECT column_name(s) FROM table1
UNION ALL
SELECT column_name(s) FROM table2;
```

2) INTERSECT

The SQL INTERSECT clause/operator is used to combine two SELECT statements, but
returns rows only from the first SELECT statement that are identical to a row in the second
SELECT statement. This means INTERSECT returns only common rows

returned by the two
SELECT statements.
Just as with the UNION operator, the same rules apply when using the INTERSECT
operator. MySQL does not support INTERSECT operator

3) EXCEPT

The SQL EXCEPT clause/operator is used to combine two SELECT statements and returns
rows from the first SELECT statement that are not returned by the second SELECT
statement. This means EXCEPT returns only rows, which are not available in second
SELECT statement.

Just as with the UNION operator, the same rules apply when using the EXCEPT operator.
MySQL does not support EXCEPT operator.

- **JOINS**

INNER JOIN

The INNER JOIN keyword selects all rows from both tables as long as there is a match
between the columns in both tables.
SQL INNER JOIN Syntax

```
SELECT column_name(s)
FROM table1
INNER JOIN table2
ON    table1.column_name=table2.column_name;
```

LEFT JOIN
The LEFT JOIN keyword returns all rows from the left table (table1), with the matching rows

in the right table (table2). The result is NULL in the right side when there is no match.

SQL LEFT JOIN Syntax

```
SELECT column_name(s)
FROM table1
LEFT JOIN table2
ON    table1.column_name=table2.column_name;
```

RIGHT JOIN

The RIGHT JOIN keyword returns all rows from the right table (table2), with the matching
rows in the left table (table1). The result is NULL in the left side when there is no match.

SQL RIGHT JOIN Syntax

```
SELECT column_name(s)
FROM table1
RIGHT JOIN table2
ON    table1.column_name=table2.column_name;
```

Examples-

1) Retrieve customer names having loan or account or both by eliminating duplicates

```
select cust_name
from borrower
union
select cust_name
from depositor;
```

2) Retrieve customer names having loan or account or both without eliminating duplicates

```
select cust_name
from borrower
```

union all
select cust_name
from depositor;

3) List customers who have placed order. select cname, order_number
from customer join order1 on order1.id=customer.customer_id;

4) List all customers whether they placed order or not. select cname, order_number
from customer left outer join order1 on order1.id=customer.customer_id;

5) List all order numbers whether they are ordered by customers or not. select cname, order_number
from customer right outer join order1 on order1.id=customer.customer_id;

Conclusion: Various queries are executed with set operations and all types of joins.

```
mysql> select lastName,firstName,stuID
    -> from student
    -> where stuID IN (select stuID from enroll where classnumber IN (select classnumber from class where facID='S100')) ORDER BY lastName,firstName ASC;
+----------+-----------+--------+
| lastName | firstName | stuID  |
+----------+-----------+--------+
| SANJU    | FOURUM    | 0074AB |
+----------+-----------+--------+
1 row in set (0.00 sec)
```

```
mysql> select facID from faculty where department='MATHS' UNION SELECT facID from class where room='6CR1';
+-------+
| facID |
+-------+
| F100  |
| S103  |
| S100  |
+-------+
3 rows in set (0.00 sec)
```

```
mysql> select count(DISTINCT stuID) from enroll where classNumber ='PROD2B';
+------------------------+
| count(DISTINCT stuID) |
+------------------------+
|                      1 |
+------------------------+
1 row in set (0.00 sec)
```

```
mysql> select stuID, 'Number of courses=',credits/3 from student;
+--------+--------------------+-----------+
| stuID  | Number of courses= | credits/3 |
+--------+--------------------+-----------+
| 0073AA | Number of courses= |    1.3333 |
| 0073AB | Number of courses= |    1.3333 |
| 0074AA | Number of courses= |    1.6667 |
| 0074AB | Number of courses= |    1.6667 |
| 0075AA | Number of courses= |    2.0000 |
| 0075AB | Number of courses= |    1.0000 |
| 0076AA | Number of courses= |    1.0000 |
| 0076AB | Number of courses= |    0.6667 |
| 0077AA | Number of courses= |    0.6667 |
| 0077AB | Number of courses= |    2.0000 |
+--------+--------------------+-----------+
10 rows in set (0.00 sec)
```

```
mysql> select classNumber, COUNT(*) from enroll group by classNumber;
+-------------+----------+
| classNumber | COUNT(*) |
+-------------+----------+
| CHEM1B      |        1 |
| DBMS2A      |        1 |
| DE3A        |        1 |
| DIS2A       |        1 |
| DS6A        |        1 |
| GRAPH2      |        1 |
| MATH1B      |        1 |
| MATH2A      |        1 |
| PHY1A       |        1 |
| PROD2B      |        1 |
+-------------+----------+
10 rows in set (0.00 sec)
```

```
mysql> select classNumber from enroll group by classNumber HAVING COUNT(*)<3;
+-------------+
| classNumber |
+-------------+
| CHEM1B      |
| DBMS2A      |
| DE3A        |
| DIS2A       |
| DS6A        |
| GRAPH2      |
| MATH1B      |
| MATH2A      |
| PHY1A       |
| PROD2B      |
+-------------+
10 rows in set (0.00 sec)

mysql> select * from class where classNumber LIKE 'MATH%';
+-------------+-------+----------+------+
| classNumber | facID | schedule | room |
+-------------+-------+----------+------+
| MATH1B      | F100  | MWTF     | 22   |
| MATH2A      | S103  | MT2WF    | 6CR1 |
+-------------+-------+----------+------+
2 rows in set (0.00 sec)
```

Name	Prerna Sunil Jadhav
SAP ID	60004220127
Experiment No.	07
Aim: Create Views and Triggers.	

Theory: A view can contain all rows of a table or select rows from a table. A view can be created from one or many tables which depend on the written SQL query to create a view. Views, which are kind of virtual tables, allow users to do the following:

- Structure data in a way that users or classes of users find natural or intuitive.
- Restrict access to the data such that a user can see and (sometimes) modify exactly what they need and no more.
- Summarize data from various tables which can be used to generate reports.

Creating Views:

Database views are created using the CREATE VIEW statement. Views can be created from a single table, multiple tables, or another view. To create a view, a user must have the appropriate system privilege according to the specific implementation.

The basic CREATE VIEW syntax is as follows:

```
CREATE VIEW
view_name AS SELECT column1,
column2.....
FROM table_name WHERE
[condition];
```

1) **create a view to select employee name and address as well as the department details in which he is working group by department number.**

```
create view emp_temp
as select ename, address, salary,
```

dno,dname from employee, department
where dno=dnumber
group by dno;

2) Display all records of a view.

Select * from emp_temp;

3) Delete the view emp_temp.

Drop view emp_temp

Trigger:

A trigger is a set of actions that are run automatically when a specified change operation (SQL INSERT, UPDATE, or DELETE statement) is performed on a specified table. Triggers are useful for tasks such as enforcing business rules, validating input data, and keeping an audit trail.

A trigger is a named database object that is associated with a table, and it activates when a particular event (e.g. an insert, update or delete) occurs for the table. The statement CREATE TRIGGER creates a new trigger in MySQL.

Syntax

```
CREATE
[DEFINER = { user | CURRENT_USER }]
TRIGGER
trigger_name trigger_time
trigger_event
ON tbl_name FOR EACH ROW trigger_body
trigger_time: { BEFORE | AFTER }
trigger_event: { INSERT | UPDATE
```

| DELETE }

Example

1) delimiter //
create trigger
depocheck before
insert on depositor
FOR EACH ROW
IF NEW.salary is null
THEN
SET NEW.salary = 5000; END IF;
//

2) delimiter //
create trigger
feed_depositor_bkp after
insert on depositor
FOR EACH ROW
insert into depositor_bkp(cust_name,salary,branch)
values (NEW.cust_name, new.salary,new.branch);
//

3) delimiter //

```
create trigger employeetrig
before update on employee
for each row
if new.dno is null then
set new.dno= 01;
end if;
//
```

4) delimiter //
create trigger
total_sal1 after
insert on
employee for
each row

```
if new.dno is not null
then update
department
set total_sal=total_sal+new.
salary where
dnumber=new.dno;
end if;
//
```

5) drop trigger feed_depositor_bkp;

CONCLUSION:

Hence, we have executed views and triggers for a college database.

Name	Prerna Sunil Jadhav
SAP ID	60004220127
Experiment No.	08
Aim: Implementations locks in various modes for concurrency control.	

Theory: A transaction is a logical unit of processing in a DBMS which entails one or more database access operation. In a nutshell, database transactions represent real-world events of any enterprise.

All types of database access operation which are held between the beginning and end transaction statements are considered as a single logical transaction. During the transaction the database is inconsistent. Only once the database is committed the state is changed from one consistent state to another.

Concurrency control is the procedure in DBMS for managing simultaneous operations without conflicting with each another. Concurrent access is quite easy if all users are just reading data. There is no way they can interfere with one another. Though for any practical database, would have a mix of reading and WRITE operations and hence the concurrency is a challenge.

Concurrency control is used to address such conflicts which mostly occur with a multi-user system. It helps you to make sure that database transactions are performed concurrently without violating the data integrity of respective databases.

Therefore, concurrency control is a most important element for the proper functioning of a system where two or multiple database transactions that require access to the same data, are executed simultaneously.

Concurrency Control Protocols

Different concurrency control protocols offer different benefits between the

amount of concurrency they allow and the amount of overhead that they impose.

- Lock-Based Protocols
- Two Phase
- Timestamp-Based Protocols
- Validation-Based Protocols

Concurrency control using Mysql Locks:

Deadlock is a state of a database system having two or more transactions, when each transaction is waiting for a data item that is being locked by some other transaction. A deadlock can be indicated by a cycle in the wait-for-graph. This is a directed graph in which the vertices denote transactions and the edges denote waits for data items.

A database locks serve to protect shared resources or objects. These protected resources could be:

* Tables
* Data Rows
* Data blocks
* Cached Items
* Connections
* Entire Systems

The deadlock detection and removal approach runs a deadlock detection algorithm periodically and removes deadlock in case there is one. It does not

check for deadlock when a transaction places a request for a lock. When a transaction requests a lock, the lock manager checks whether it is available. If it is available, the transaction is allowed to lock the data item; otherwise the transaction is allowed to wait.

Since there are no precautions while granting lock requests, some of the transactions may be deadlocked. To detect deadlocks, the lock manager periodically checks if the wait-for graph has cycles.If the system is deadlocked, the lock manager chooses a victim transaction from each cycle. The victim is aborted and rolled back; and then restarted later. Some of the methods used for victim selection are –

* Choose the youngest transaction.

* Choose the transaction with fewest data items.

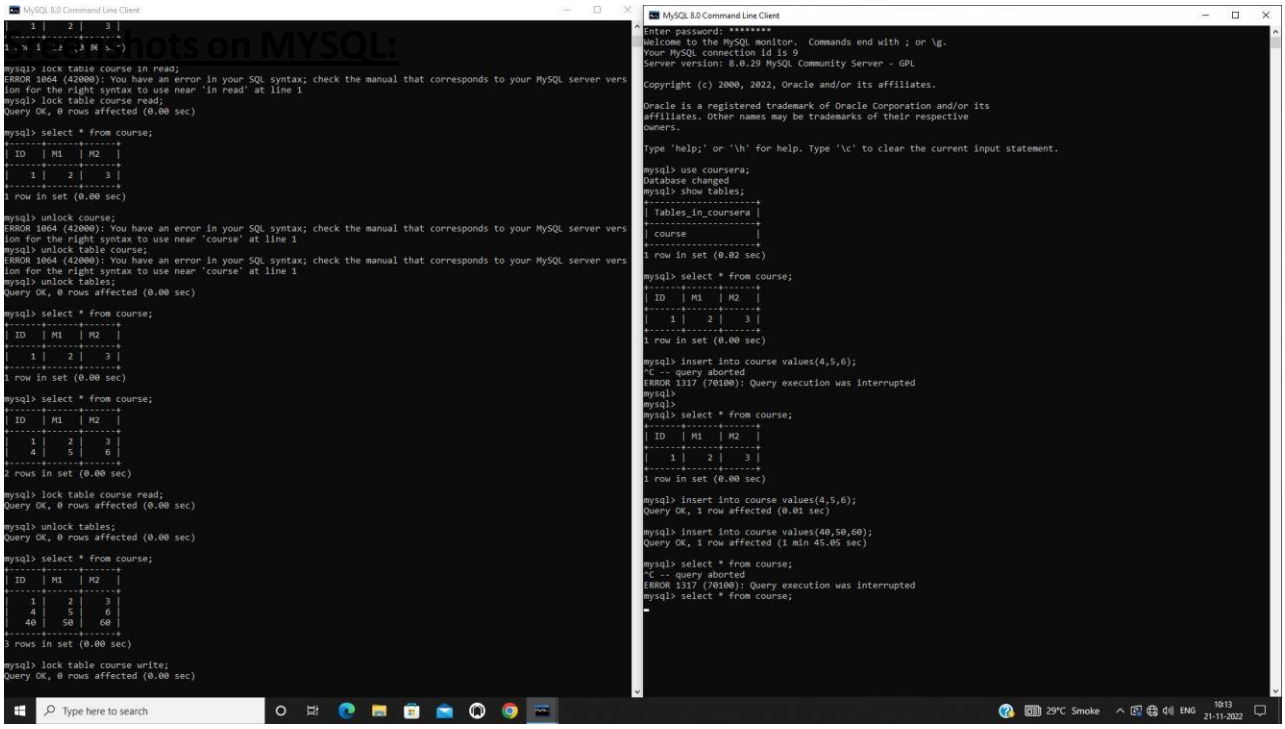
* Choose the transaction that has performed least number of updates.

* Choose the transaction having least restart overhead.

* Choose the transaction which is common to two or more cycles.

This approach is primarily suited for systems having transactions low and where fast response to lock requests is needed.

Screenshots on MYSQL:



Name	Prerna Sunil Jadhav
SAP ID	60004220127
Experiment No.	09
Aim: Case study on Fragmentation (PHf, DHF, Vf and Hf)	

Theory: Networks of computers are found in everywhere. The Internet is one, as are the many networks of which it is composed. Mobile phone networks, corporate networks, factory networks, campus networks, home networks, in-car networks _ all of these type of networks, both separately and in combination, share the essential characteristics that make them relevant subjects for study under the heading distributed systems.

Distributed systems are an important development in computing technology, which is concerned with the delivery of constantly expanding data to points of query. Collections of data in the forms of partitions or fragments can be distributed or replicated over multiple physical locations. When an organization is geographically dispersed, it may choose to store its databases on a central database server or to distribute them to local servers (or a combination of both).

A **distributed database** is a single logical database that is spread physically across computers in multiple locations that are connected by a data communications network. A distributed database (DDB) is a collection of data that logically belongs to the same system but is spread over the sites of a computer network. Distribution of data is a collection of fragmentation, replication and allocation processes. The sites of the distributed database may be allocated in the same space and have the same network address but the communication between them is done over a network instead of sharing memory.

As communication technology, software and hardware, advance rapidly and the equipment prices falls every day, developing DDBS become more and more feasible. Design of efficient distributed database is one of the hot topics in database and information technology areas

Fragmentations

Distributed database system design primary concern is to make fragmentation of classes in case of object-oriented databases, or the relations in case of relational database, replication and allocation of the fragments in different sites of the distributed system, and local optimization in each site. Fragmentation is a technique to split a single

class or relation of a database into two or more partitions, also; the combination of the partitions supports the original database without any loss of information. That means, it reduces the amount of irrelevant data accessed by the applications of the database, thus reducing the number of disk accesses. Each fragment may be stored at any site over a computer network.

Fragmentation aims to improve:

1. Reliability.

2. Performance.

3. Balanced storage capacity and costs.

4. Communication costs.

5. Security.

The following information is used to decide fragmentation: Quantitative information: cardinality of relations, frequency of queries, site where query is run, selectivity of the queries, etc.

Qualitative information: predicates in queries, types of access of data, read/write, etc.

Fragmentation Types

Fragmentation can be horizontal, vertical or mixed.

1. Horizontal Fragmentation

Horizontal fragmentation (HF) allows a relation or class to be partitioned into disjoint tuples or instances. Each fragment is stored at a different node, and each fragment has unique rows. However, the unique rows all have the same attributes (columns). Intuition behind horizontal fragmentation is that Every site should hold all information that is used to query at the site and the information at the site should be fragmented so the queries of the site run faster.

2. Vertical Fragmentation(VF)

A class or relation in VF will be partitioned into separate sets of columns or attributes except the primary key. Each set must include the primary key attribute(s) of the table. This arrangement can make sense when different sites are responsible for processing different functions involving an entity.

The main target of vertical fragmentation is to divide a relation into a set of smaller relations, so that in only one fragment many of the applications will run.

3. Mixed Fragmentation

Combination of Horizontal and vertical fragmentations give the mixed fragmentations (MF). The table in this type of fragmentation scheme is divided into blocks that based on the needed requirements. Each fragmentation may be allocated on a specific site. Mixed fragmentation is the most complex one; it needs more management. Mixed fragmentation contains a vertical fragment followed schema will not be sufficient to satisfy the requirements by a vertical fragmentation, or a horizontal fragmentation followed by a vertical fragmentation.

Correctness Rules of Fragmentation

1. Completeness

Decomposition of relation R into fragments R1,R2,…,Rn is complete iff each data item in R can also be found in some Ri.

2. Reconstruction

If relation R is decomposed into fragments R1, R2,…, Rn, then there should exist some relational operator ∇that reconstructs R from its fragments, i.e., R = R1∇…∇Rn:

Union to combines horizontal fragments.

Join to combine vertical fragments.

3. Disjointness

If relation R is decomposed into fragments R1, R2,…, Rn and data item di appears in fragment Rj, then di should not appear in any other fragment Rk, k = j (exception: primary key attribute for vertical fragmentation

for horizontal fragmentation, data item is a tuple.

for vertical fragmentation, data item is an attribute.

For each fragment of a relation R, the comparison of fragmentation strategies is shown.

1. Condition C = True (all tuples are selected).

2. List (L = ATTRS(R)) = True (all attributes are included in the list).

Name	Prerna Sunil Jadhav
SAP ID	60004220127
Experiment No.	10
Aim: Case study on Recent Databases and applications.	

Theory: Historically, Database Management systems (DBMS) were simple software programs and associated hardware that allowed users to access data from different geographical locations. The system offers its users the ability to store data without concerns about structural changes, or the data's physical location. Additionally, a Database Management system (DBMS) can set restrictions on the data being used, and the services available to each user.

DBMSs are changing, however. They are expanding, taking on more responsibilities, and providing smarter answers. As new goals and problems present themselves, the desire to find new ways to use Database Management systems prompt unique solutions. Many of these innovations are available only in cloud-based DBMSs. As Database Management systems develop new features and new options, it makes sense to re- examine the organization's current system, and consider all new options.

The market for Database Management systems is growing fast and, according to Research and Markets, the global DBMS market was estimated to have reached $63.9 trillion in 2020, and is projected to reach
$142.7 trillion by 2027.

Increasingly, organizations are merging their data warehouses and data lakes into cloud storage systems. Shifting to the cloud requires a Database Management system (DBMS) for working with a broad range of new data formats.

Database Management trends in 2022 include:

- Cloud-based DBMS
- Automation and DBMS
- Augmented DBMS
- Increased security
- In-memory databases
- Graph databases
- Open source DBMSs
- Databases-as-a-service

These trends are based, to a large extent, on businesses wanting to provide access to their products and services over the internet, with the goal of maintaining (or increasing) profits during the pandemic.

- **Netflix Database Study:**

In 2019 Netflix began to run into problems with Cassandra for certain use cases. At that time Netflix was seeing an increase in demand and an increase in their database requirements. For example, the studio side of Netflix was producing more shows and movies which added new data needs. Generally speaking, there was an increase in the need for consistent data, specifically for these three use cases:

• Cloud drive service - a file system-like service for media assets which was needed by the Netflix studio side of the business.

• Content delivery - Netflix built its own CDN called Open Connect, and they needed a control plane service to manage network devices around the world.

• Spinnaker - a continuous delivery platform on the cloud.

These are all global services that need to support consistent transactions at times. Supporting consistent transactions is problematic with Cassandra, because in Cassandra you don't get rich transactions. Instead, you have lightweight transactions, which are extremely performant, but limited. Also, the secondary indices in Cassandra are unreliable and often

don't work.
The other options Netflix considered for these use cases were AWS Aurora as well as DynamoDB but they ran into some limitations with scalability there.

Netflix decided that they needed a scalable SQL database for those three use cases and established a set of requirements: Multi-active topology, Global consistent secondary indices, global transactions, open source, and SQL.

In 2020 Netflix deployed their first CockroachDB cluster in production. Today they have 100 production clusters and 150+ test clusters. At this time most of the clusters are deployed in a single region with three availability zones. The biggest CockroachDB cluster at Netflix is a 60-node, single-region cluster with 26.5 terabytes of data.

In this section of the presentation, Shengwei Wang explains why Netflix does not use the CockroachDB binary, how Netflix navigates upgrades to new versions of CockroachDB, and how Netflix deploys/manages CockroachDB.

Conclusion: Hence it is evident from the above case study the importance of Database Management Systems in the modern world along with the developments that have taken place in the recent years.

THANK YOU