Database Management System

(CSL 214)

Lab Workbook

Faculty Name: Ms. Anuradha Dhull

Student Name: Piyush Gambhir

Roll No.: 21CSU349

Semester: IV

Group: AL-3

Department of Computer Science and Engineering

The NorthCap University, Gurugram- 122017, India

Session 2022-23

AL-3

Student Name: Piyush Gambhir

Student Roll No.: 21CSU349

INDEX

S.No	Experiment Title	Date of Experiment	Date of Submission	Sign Student	CO Covered	Sign Faculty
1	Design an ER/EER diagram for the COMPANY and SPORTS TEAM database.	20.01.2023	06.02.2023	Piyush	CO2	06/02/23
2	Design a Relational Database Schema for the COMPANY and SPORTS TEAM database from the ER/EER diagram.	30.01.2023	06.02.2023	Piyush	CO2	06/2/23
3	To apply SQL integrity constraints as per the DDL statements given below for COMPANY database.	23.01.2023	30.01.2023	Piyush	CO4	30/1/23
4	To familiarize with SELECT-FROM-WHERE SQL simple queries on the COMPANY database.	13.02.2023	20.02.2023	Piyush	CO4	20/02/23
5	To familiarize with JOIN operations in SQL on the COMPANY database.	27.02.2023	06.03.2023	Piyush	CO4	06/03/23
6	To understand Aggregate functions using SQL queries on the COMPANY database.	13.03.2023	13.03.2023	Piyush	CO4	13/03/23
7	To familiarize with nested SQL queries on the COMPANY database.	03.04.2023	03.04.2023	Piyush	CO4	03/04/23
8	Identifying contrast between Relational Databases and NoSQL, thereby recognizing their applications.	17.04.2023	24.04.2023	Piyush	CO5	24/04/23
9	Create COMPANY database using NoSQL database - MongoDB.	17.04.2023	24.04.2023	Piyush	CO5	24/04/23
10	Retrieve data from NoSQL database - MongoDB.	24.04.2023	24.04.2023	Piyush	CO5	24/04/23
11	VA - Design an ER/ EER Diagram, relational schema and implement the database in MongoDB.				CO2, CO5	

Experiment No: 1

Student Name and Roll Number: Piyush Gambhir – 21CSU349
Semester /Section: Semester-IV – AIML-B (AL-3)
Link to Code: Piyush-Gambhir/NCU-CSL214-DBMS-Lab_Manual (github.com)
Date: 30.01.2023
Faculty Signature:
Marks:

Objective

Design an ER/EER diagram for the COMPANY and SPORTS TEAM database.

Program Outcome

- The students will be able to draw conceptual database design using ERD Plus.

Problem Statement

1. The COMPANY database keeps track of a company's employees, departments, and projects. Suppose that after the requirements collection and analysis phase, the database designers provide the following description of the mini world—the part of the company that will be represented in the database.

- A department controls several projects, each of which has a unique name, a unique number, and a single location.
- The company is organized into departments. Each department has a unique name, a unique number, and a particular employee who manages the department. We keep track of the start date when that employee began managing the department. A department may have several locations.
- We store each employee's name, Social Security number, address, salary, sex (gender), and birth date. An employee is assigned to one department, but may work on several projects, which are not necessarily controlled by the same department. We keep track of the current number of hours per week

that an employee works on each project. We also keep track of the direct supervisor of each employee (who is another employee).

- We want to keep track of the dependents of each employee for insurance purposes. We keep each dependent's first name, sex, birth date, and relationship to the employee. `

2. Design an ER/EER diagram for keeping track of the exploits of your favourite sports team. You should store the matches played, the scores in each match, the players in each match and individual player statistics for each match. Summary statistics should be modelled as derived attributes. Further, extend the E-R diagram of the previous question to track the same information for all teams in a league.

Design the ER/EER model by identifying the following from the above requirements:

i) Entities (Strong and Weak)
ii) Relationships
iii) Participation constraints
iv) Various types of attributes
v) Recursive relations
vi) Mapping cardinalities
vii) Binary/Ternary relationship
viii) Specialization/Generalization etc.

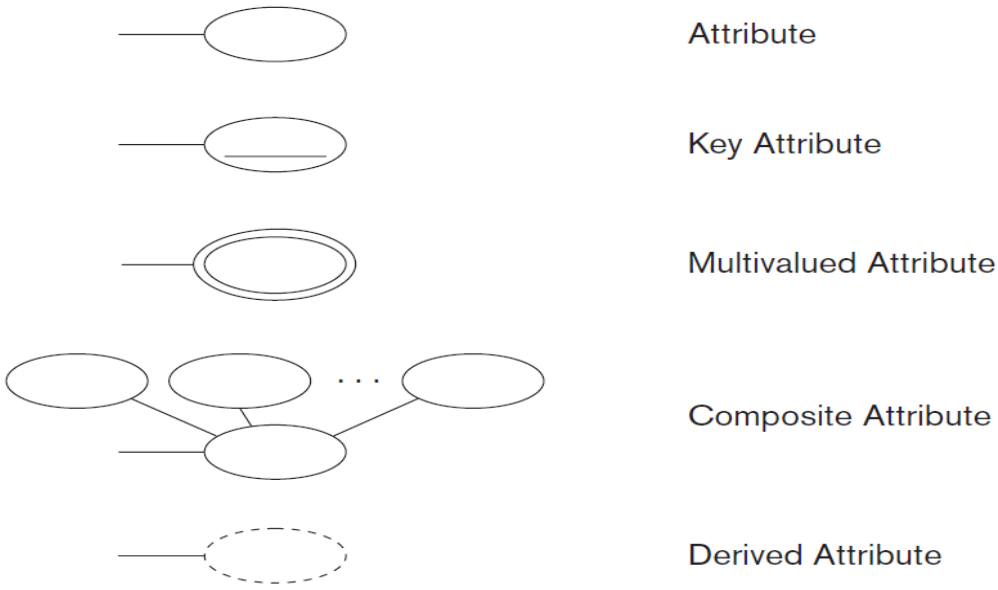
Background Study

ER Diagram Symbols and Notations:

1) Entity:
   - Real-world object distinguishable from other objects.
   - An entity is described using a set of *attributes*.

2) Entity Set: A collection of similar entities. Eg: all employees.
   - All entities in an entity set have the same set of attributes.
   - Each entity set has a *key*.
   - Each attribute has a *domain*.

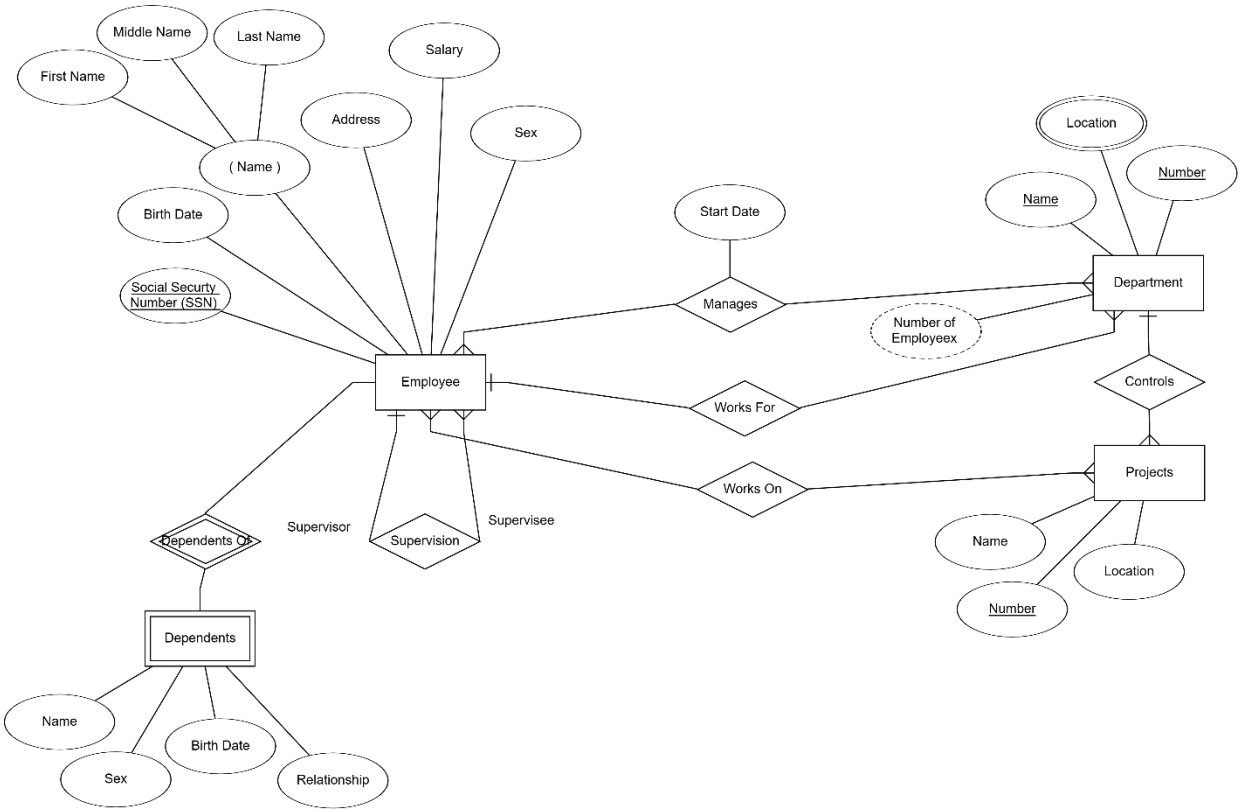
3) Attributes
    - Attributes are properties used to describe an entity.
    - Example: EMPLOYEE entity may have a Name, SSN, Address, Sex, BirthDate.



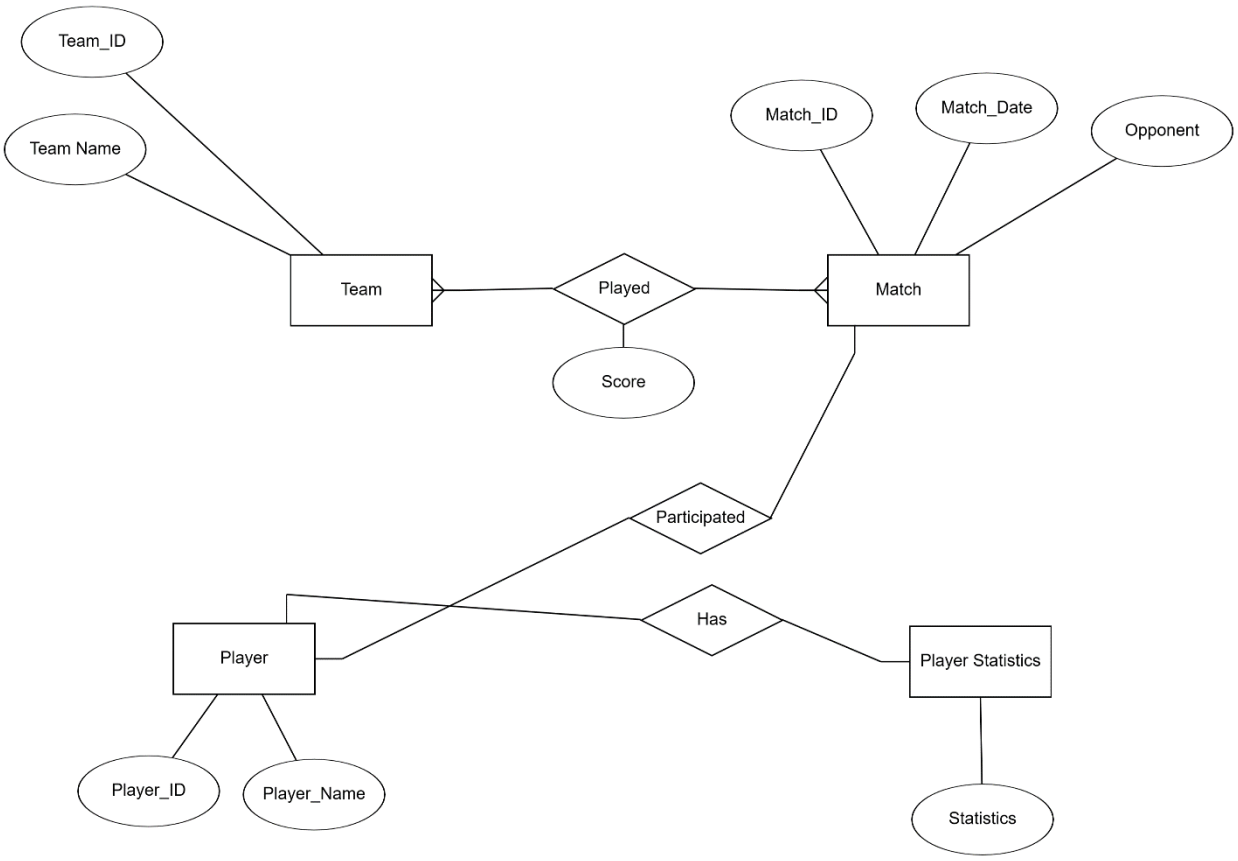
4) Relationship
    - A relationship relates two or more distinct entities with a specific meaning.
    - Relationships of the same type are grouped or typed into a relationship type.
    - 3 types of relationships : Unary, Binary & Ternary.
5) Recursive Relationship
    - A relationship with the same participating entity type in distinct roles.
    - Example: the SUPERVISION relationship
6) Structural Constraints – Semantics of Relationships
    - Cardinality Ratio : The number of instances of an entity from a relation that can be associated with the relation.
    - Participation Constraints
        - Total Participation − Each entity is involved in the relationship.
        - Partial participation − Not all entities are involved in the relationship.

ER Diagram

Company Database



Sports Team Database



Preparatory Questions

Q1) Given the basic ER and relational models, which of the following is INCORRECT?

1) An attribute of an entity can have more than one value.
2) An attribute of an entity can be composite.
3) In a row of a relational table, an attribute can have more than one value.
4) In a row of a relational table, an attribute can have exactly one value or a NULL value.

Q2) Consider a directed line(-->) from the relationship set advisor to both entity sets instructor and student. This indicates _____ cardinality

1) One to many
2) One to one
3) Many to many
4) Many to one

Q3) An entity set that does not have sufficient attributes to form a primary key is termed as:

1) Strong entity set
2) Variant set
3) Weak entity set
4) Variable set

Q4) Which of the following indicates the maximum number of entities can be involved in a relationship?

1) Minimum cardinality
2) Maximum Cardinality
3) ERD
4) Greater Entity Count (GEC)

Q5) State true or false: Every weak entity must be associated with and identifying entity.

1) True
2) False

Experiment No: 2

Student Name and Roll Number: Piyush Gambhir – 21CSU349
Semester /Section: Semester-IV – AIML-B (AL-3)
Link to Code: Piyush-Gambhir/NCU-CSL214-DBMS-Lab_Manual (github.com)
Date:
Faculty Signature:
Marks:

Objective

Design a Relational Database Schema for the COMPANY and SPORTS TEAM database from the ER/EER diagram.

Program Outcome

- The students will be able to map the conceptual database design to logical (relational) database design.

Problem Statement

Map the ER/EER diagram of the COMPANY and SPORTS TEAM database created in the previous experiment to Relational Database Schema. Follow the below mentioned steps to successfully map ER/EER model to relational tables:

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
Step 8: Mapping EER Model Constructs to Relations
Step 9: Options for Mapping Specialization or Generalization

Step 10: Mapping of Union Types (Categories)

Background Study

1) Mapping of Regular Entity Types
   - For each regular (strong) entity type E in the ER schema, create a relation R that includes all the simple attributes of E. Choose one of the key attributes of E as the primary key for R. If the chosen key of E is composite, the set of simple attributes that form it will together form the primary key of R.

2) Mapping of Weak Entity Types
   - For each weak entity type W in the ER schema with owner entity type E, create a relation R and include all simple attributes (or simple components of composite attributes) of W as attributes of R.
   - Include as foreign key attributes of R the primary key attribute(s) of the relation(s) that correspond to the owner entity type(s).

3) Mapping of Binary 1:1 Relationship Types
   - For each binary 1:1 relationship type R in the ER schema, identify the relations S and T that correspond to the entity types participating in R. This has 3 approaches :
     - Foreign Key Approach
     - Merged Relation Option
     - Cross-reference or Relationship Relation Option

4) Mapping of Binary 1:N Relationship Types
   - For each regular binary 1:N relationship type R, identify the relation S that represent the participating entity type at the N-side of the relationship type.
   - Include as foreign key in S the primary key of the relation T that represents the other entity type participating in R.

5) Mapping of Binary M:N Relationship Types
   - For each regular binary M:N relationship type R, create a new relation S to represent R.
   - Include as foreign key attributes in S the primary keys of the relations that represent the participating entity types; their combination will form the primary key of S.

6) Mapping of Multi-valued attributes

- For each multi-valued attribute, A, create a new relation R. This relation R will include an attribute corresponding to A, plus the primary key attribute K-as a foreign key in R-of the relation that represents the entity type of relationship type that has A as an attribute.

7) Mapping of N-ary Relationship Types
   - For each n-ary relationship type R, where n>2, create a new relationship S to represent R.
   - Include as foreign key attributes in S the primary keys of the relations that represent the participating entity types.

8) Options for Mapping Specialization or Generalization
   - Convert each specialization with m subclasses $\{S_1, S_2,\ldots,S_m\}$ and generalized superclass C, where the attributes of C are $\{k,a_1,\ldots a_n\}$ and k is the (primary) key, into relational schemas using 1 of the following :
     1) Multiple relations-Superclass and subclasses.
     2) Multiple relations-Subclass relations only :
     3) Single relation with *one type attribute*
     4) Single relation with multiple type attributes.

9) Mapping of Union Types (Categories)
   - For mapping a category whose defining superclass have different keys, it is customary to specify a new key attribute, called a surrogate key, when creating a relation to correspond to the category.

Relational Database Design

Company Database

Experiment - 2

Company

1. Employee (Fname, Minit, Lname, Ssn, Bdate, Address, Sex, Salary, SuperSsn, Dno)

2. Department (Dname, Dnumber, MgrSsn, Startdate)

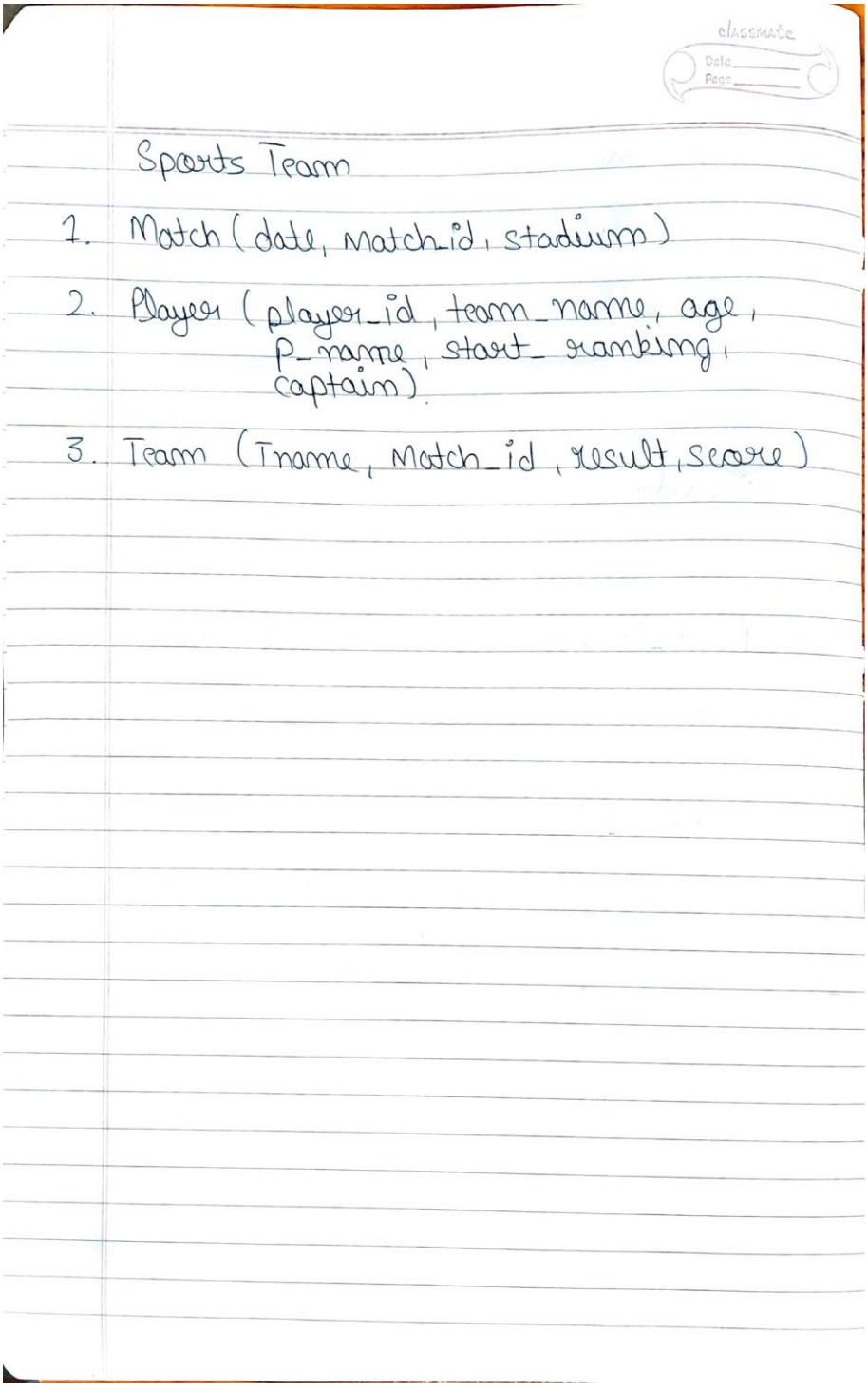
3. Dept_Location (Dnumber, Dlocation)

4. Project (Pname, Pnumber, Plocation, Dnum)

5. Works_on (Ssn, Pnumber, Hours)

6. Dependent (Ssn, Name, DOB, Gender, Relationship)

Sports Team Database

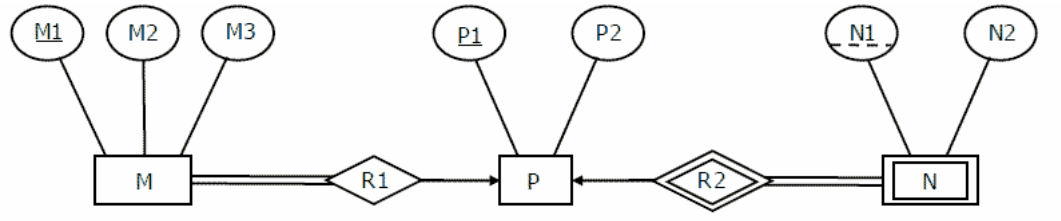


Preparatory Questions

Q1) In which of the following, a separate schema is created consisting of that attribute and the primary key of the entity set.

1) A many-to-many relationship set
2) A multivalued attribute of an entity set
3) A one-to-many relationship set
4) All of the mentioned

Q2) Consider the following ER diagram.



The minimum number of tables needed to represent M, N, P, R1, R2 is

1) 2
2) 3
3) 4
4) 5

Q3) Consider the data given in above question. Which of the following is a correct attribute set for one of the tables for the correct answer to the above question?

1) {M1, M2, M3, P1}
2) {M1, P1, N1, N2}
3) {M1, P1, N1}
4) {M1, P1}

Q4) Let E1 and E2 be two entities in an E/R diagram with simple single-valued attributes. R1 and R2 are two relationships between E1 and E2, where R1 is one-to-many and R2 is many-to-many. R1 and R2 do not have any attributes of their own. What is the minimum number of tables required to represent this situation in the relational model?

1) 2
2) 3
3) 4

4) 5

Q5) What is the min and max number of tables required to convert an ER diagram with 2 entities and 1 relationship between them with partial participation constraints of both entities?

1) Min 1 and max 2
2) Min 1 and max 3
3) Min 2 and max 3
4) Min 2 and max 2

Experiment No: 3

Student Name and Roll Number: Piyush Gambhir – 21CSU349
Semester /Section: Semester-IV – AIML-B (AL-3)
Link to Code: Piyush-Gambhir/NCU-CSL214-DBMS-Lab_Manual (github.com)
Date: 23.01.2023
Faculty Signature:
Marks:

Objective

To apply SQL integrity constraints as per the DDL statements given below for COMPANY database.

Program Outcome

- The students will understand how a database is created followed by insertion of relevant data.
- The students will understand the need of applying various types of integrity constraints such as primary key, foreign key, unique key, NOT NULL, default and CHECK etc

Problem Statement

```
CREATE TABLE EMPLOYEE
        ( Fname          VARCHAR(15)          NOT NULL,
          Minit          CHAR,
          Lname          VARCHAR(15)          NOT NULL,
          Ssn            CHAR(9)              NOT NULL,
          Bdate          DATE,
          Address        VARCHAR(30),
          Sex            CHAR,
          Salary         DECIMAL(10,2),
          Super_ssn      CHAR(9),
          Dno            INT                  NOT NULL,
        PRIMARY KEY (Ssn),
        FOREIGN KEY (Super_ssn) REFERENCES EMPLOYEE(Ssn),
        FOREIGN KEY (Dno) REFERENCES DEPARTMENT(Dnumber) );
```

```
CREATE TABLE DEPARTMENT
        ( Dname                 VARCHAR(15)             NOT NULL,
          Dnumber               INT                     NOT NULL,
          Mgr_ssn               CHAR(9)                 NOT NULL,
          Mgr_start_date        DATE,
        PRIMARY KEY (Dnumber),
        UNIQUE (Dname),
        FOREIGN KEY (Mgr_ssn) REFERENCES EMPLOYEE(Ssn) );

CREATE TABLE DEPT_LOCATIONS
        ( Dnumber               INT                     NOT NULL,
          Dlocation             VARCHAR(15)             NOT NULL,
        PRIMARY KEY (Dnumber, Dlocation),
        FOREIGN KEY (Dnumber) REFERENCES DEPARTMENT(Dnumber) );
CREATE TABLE PROJECT
        ( Pname                 VARCHAR(15)             NOT NULL,
          Pnumber               INT                     NOT NULL,
          Plocation             VARCHAR(15),
          Dnum                  INT                     NOT NULL,
        PRIMARY KEY (Pnumber),
        UNIQUE (Pname),
        FOREIGN KEY (Dnum) REFERENCES DEPARTMENT(Dnumber) );
CREATE TABLE WORKS_ON
        ( Essn                  CHAR(9)                 NOT NULL,
          Pno                   INT                     NOT NULL,
          Hours                 DECIMAL(3,1)            NOT NULL,
        PRIMARY KEY (Essn, Pno),
        FOREIGN KEY (Essn) REFERENCES EMPLOYEE(Ssn),
        FOREIGN KEY (Pno) REFERENCES PROJECT(Pnumber) );
CREATE TABLE DEPENDENT
        ( Essn                  CHAR(9)                 NOT NULL,
          Dependent_name        VARCHAR(15)             NOT NULL,
          Sex                   CHAR,
          Bdate                 DATE,
          Relationship          VARCHAR(8),
        PRIMARY KEY (Essn, Dependent_name),
        FOREIGN KEY (Essn) REFERENCES EMPLOYEE(Ssn) );
```

Implement the following types of integrity constraints:

1) Primary Key
2) Foreign Key
3) Unique
4) Default
5) Auto-increment
6) Check
7) Not Null

Background Study

1) Primary Key Constraint: 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.

Syntax to define a Primary key at column level:
column name datatype [CONSTRAINT constraint_name] PRIMARY KEY

Syntax to define a Primary key at table level:
[CONSTRAINT constraint_name] PRIMARY KEY (column_name1,column_name2,..)

Rules for defining Primary key:

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

2) Foreign Key (Referential integrity constraint): This constraint identifies any column referencing the PRIMARY KEY in another table. It establishes a relationship between two columns in the same table or between different tables. For a column to be defined as a Foreign Key, it should be a defined as a Primary Key in the table which it is referring. One or more columns can be defined as Foreign key.

Syntax to define a Foreign key at column level:
[CONSTRAINT constraint_name] REFERENCES Referenced_Table_name(column_name)

Syntax to define a Foreign key at table level:
[CONSTRAINT constraint_name] FOREIGN KEY(column_name) REFERENCES referenced_table_name(column_name);

3) SQL Not Null Constraint : This constraint ensures all rows in the table contain a definite value for the column which is specified as not null. Which means a null value is not allowed.

Syntax to define a Not Null constraint:
[CONSTRAINT constraint name] NOT NULL

4) SQL Unique Key: This constraint ensures that a column or a group of columns in each row have a distinct value. A column(s) can have a null value but the values cannot be duplicated.

Syntax to define a Unique key at column level:
[CONSTRAINT constraint_name] UNIQUE

Syntax to define a Unique key at table level:
[CONSTRAINT constraint_name] UNIQUE(column_name)

5) SQL Check Constraint : This constraint defines a business rule on a column. All the rows must satisfy this rule. The constraint can be applied for a single column or a group of columns.
Syntax to define a Check constraint:
[CONSTRAINT constraint_name] CHECK (condition)

Output: Screenshots

Create Table Department

```
mysql> CREATE TABLE DEPARTMENT(
    ->     Dname VARCHAR(15) NOT NULL,
    ->     Dnumber INT NOT NULL,
    ->     Mgr_ssn CHAR(9) NOT NULL,
    ->     Mgr_start_date DATE,
    ->     PRIMARY KEY(Dnumber),
    ->     UNIQUE(Dname)
    -> );
Query OK, 0 rows affected (0.02 sec)
```

Create Table Employee

```
mysql> CREATE TABLE EMPLOYEE(
    ->     Fname VARCHAR(15) NOT NULL,
    ->     Minit CHAR,
    ->     Lname VARCHAR(15) NOT NULL,
    ->     Ssn CHAR(9) NOT NULL,
    ->     Bdate DATE,
    ->     Address VARCHAR(30),
    ->     Sex CHAR,
    ->     Salary DECIMAL(10, 2),
    ->     Super_ssn CHAR(9),
    ->     Dno INT NOT NULL,
    ->     PRIMARY KEY (Ssn),
    ->     FOREIGN KEY (Super_ssn) REFERENCES EMPLOYEE(Ssn),
    ->     FOREIGN KEY (Dno) REFERENCES DEPARTMENT(Dnumber)
    -> );
Query OK, 0 rows affected (0.05 sec)

mysql> ALTER TABLE DEPARTMENT
    -> ADD FOREIGN KEY(Mgr_ssn) REFERENCES EMPLOYEE(Ssn);
Query OK, 0 rows affected (0.06 sec)
```

Create Table Department Locations

```
mysql> CREATE TABLE DEPARTMENT_LOCATIONS(
    ->     DNumber INT NOT NULL,
    ->     DLocation VARCHAR(15) NOT NULL,
    ->     PRIMARY KEY(DNumber, DLocation),
    ->     FOREIGN KEY(DNumber) REFERENCES DEPARTMENT(Dnumber)
    -> );
Query OK, 0 rows affected (0.01 sec)
```

Create Table Projects

```
mysql> CREATE TABLE PROJECT(
    ->     Pname VARCHAR(15) NOT NULL,
    ->     Pnumber INT NOT NULL,
    ->     Plocation VARCHAR(15),
    ->     Dnum INT NOT NULL,
    ->     PRIMARY KEY(Pnumber),
    ->     UNIQUE(Pname),
    ->     FOREIGN KEY(Dnum) REFERENCES DEPARTMENT(Dnumber)
    -> );
Query OK, 0 rows affected (0.03 sec)
```

Create Table Works On

```
mysql> CREATE TABLE WORKS_ON(
    ->     Essn CHAR(9) NOT NULL,
    ->     Pno INT NOT NULL,
    ->     Hours DECIMAL(3, 1),
    ->     PRIMARY KEY(Essn, Pno),
    ->     FOREIGN KEY(Essn) REFERENCES EMPLOYEE(Ssn),
    ->     FOREIGN KEY(Pno) REFERENCES PROJECT(Pnumber)
    -> );
Query OK, 0 rows affected (0.02 sec)
```

Create Table Dependent

```
mysql> CREATE TABLE DEPENDENT(
    ->     Essn CHAR(9) NOT NULL,
    ->     Dependent_name VARCHAR(15) NOT NULL,
    ->     Sex CHAR,
    ->     Bdate DATE,
    ->     Relationship VARCHAR(15),
    ->     PRIMARY KEY(Essn, Dependent_name),
    ->     FOREIGN KEY(Essn) REFERENCES EMPLOYEE(Ssn)
    -> );
Query OK, 0 rows affected (0.02 sec)
```

Preparatory Questions

1) Suppose (A, B) and (C,D) are two relation schemas. Let r1 and r2 be the corresponding relation instances. B is a foreign key that refers to C in r2. If data in r1 and r2 satisfy referential integrity constraints, which of the following is ALWAYS TRUE?

(A) $\Pi_B(r_1) - \Pi_C(r_2) = \varnothing$

(B) $\Pi_C(r_2) - \Pi_B(r_1) = \varnothing$

(C) $\Pi_B(r_1) = \Pi_C(r_2)$

(D) $\Pi_B(r_1) - \Pi_C(r_2) \neq \varnothing$

1) A
2) B
3) C
4) D

2) Given the following statements:
S1: A foreign key declaration can always be replaced by an equivalent check assertion in SQL.
S2: Given the table R(a,b,c) where a and b together form the primary key, the following is a valid table definition.

```
CREATE TABLE S (
        a INTEGER,
        d INTEGER,
        e INTEGER,
        PRIMARY KEY (d),
        FOREIGN KEY (a) references R)
```

Which one of the following statements is CORRECT?
1) S1 is TRUE and S2 is FALSE.
2) Both S1 and S2 are TRUE.
3) S1 is FALSE and S2 is TRUE.
4) Both S1 and S2 are FALSE

Q3) Which of the following is not an integrity constraint?

1) Not null
2) Positive
3) Unique
4) Check 'predicate'

Q4) *CREATE TABLE Manager(ID NUMERIC,Name VARCHAR(20),budget NUMERIC,Details VARCHAR(30));*
In order to ensure that the value of budget is non-negative which of the following should be used?

1) Check(budget>0)
2) Check(budget<0)
3) Alter(budget>0)
4) Alter(budget<0)

Q5) The following table has two attributes A and C where A is the primary key and C is the foreign key referencing A with on-delete cascade.

A	C
2	4
3	4
4	3
5	2
7	2
9	5
6	4

The set of all tuples that must be additionally deleted to preserve referential integrity when the tuple (2,4) is deleted is:

1) (3,4) and (6,4)
2) (5,2) and (7,2)
3) (5,2), (7,2) and (9,5)
4) (3,4), (4,3) and (6,4)

Experiment No: 4

Student Name and Roll Number: Piyush Gambhir – 21CSU349
Semester /Section: Semester-IV – AIML-B (AL-3)
Link to Code: Piyush-Gambhir/NCU-CSL214-DBMS-Lab_Manual (github.com)
Date: 13.02.2023
Faculty Signature:
Marks:

Objective

To familiarize with SELECT-FROM-WHERE SQL simple queries on the COMPANY database.

Program Outcome

- The students will be able to retrieve zero or more rows from one or more database tables or database views.

Problem Statement

From the COMPANY database as mentioned and described in the previous database:

1) Retrieve the birth date and address of the employee(s) whose name is 'John B. Smith'.
2) Retrieve the name and address of all employees who work for the 'Research' department.
3) For every project located in 'Stafford', list the project number, the controlling department number, and the department manager's last name, address, and birth date.
4) Select all combinations of EMPLOYEE Ssn and DEPARTMENT Dname in the database.
5) Retrieve all the attribute values of any EMPLOYEE who works in DEPARTMENT number 5.
6) Retrieve all distinct salary values.

7) Make a list of all project numbers for projects that involve an employee whose last name is 'Smith', either as a worker or as a manager of the department that controls the project.
8) Retrieve all employees whose address is in Houston, Texas.
9) Find all employees who were born during the 1950s
10) Show the resulting salaries if every employee working on the 'ProductX' project is given a 10 percent raise.
11) Retrieve a list of employees and the projects they are working on, ordered by department and, within each department, ordered alphabetically by last name, then first name.

Background Study

1) Structured Query Language SQL contains statements for data definitions, queries, and updates (both DDL and DML)

2) Domain
   - Name used with the attribute specification.
   - Makes it easier to change the data type for a domain that is used by numerous attributes.

3) Inserting values in our table using the commands:
   INSERT INTO EMPLOYEE (Fname, Lname, Dno, Ssn)

   VALUES ('Richard', 'Marini', 4, '653298653')

4) Out of the complete Database we create we can easily pick out and filter certain amount of data by using the SQL queries as :

   SELECT <attribute list>

   FROM <table list>

   WHERE <condition>;

   - <attribute list> is a list of attribute names whose values are to be retrieved by the query.
   - <table list> is a list of the relation names required to process the query.
   - <condition> is a conditional (Boolean) expression that identifies the tuples to be retrieved by the query.

5) Ambiguous Attribute Names: Same name can be used for two (or more) attributes as long as the attributes are in different relations, these are made to differ by mentioning their table names before the attribute names.
6) Aliasing, Renaming Tuple variables: We can rename the tables into some smaller easier names as per choice when it has to be used multiple times in the query.

7) DISTINCT keyword is used when the query wishes to derive no two duplicate values.

8) SET operations such as UNION and INTERSECT can also be applied on the tables to filter out the values as per choice.

9) Substring Pattern matching is carried out by the use of the keyword LIKE which helps in retrieving the column values of the tuple matching our mentioned substring.

10) ORDERBY <attributes>
    - Helps in sorting the tuple values of the attributes by default set to ascending and can be changed to descending .

Output: Screenshots

```
mysql> -- USING DATABASE
mysql> USE EXPERIMENT_4;
Database changed

mysql> -- CREATING TABLES
mysql> CREATE TABLE DEPARTMENT(
    ->     Dname VARCHAR(30) NOT NULL,
    ->     Dnumber INT NOT NULL,
    ->     Mgr_ssn CHAR(9),
    ->     Mgr_start_date DATE,
    ->     PRIMARY KEY(Dnumber),
    ->     UNIQUE(Dname)
    -> );
Query OK, 0 rows affected (0.04 sec)


mysql> CREATE TABLE EMPLOYEE(
    ->     Fname VARCHAR(15) NOT NULL,
    ->     Minit CHAR,
    ->     Lname VARCHAR(15) NOT NULL,
    ->     Ssn CHAR(9) NOT NULL,
    ->     Bdate DATE,
    ->     Address VARCHAR(30),
    ->     Sex CHAR,
    ->     Salary DECIMAL(10, 2),
    ->     Super_ssn CHAR(9),
    ->     Dno INT,
    ->     PRIMARY KEY (Ssn),
    ->     FOREIGN KEY (Super_ssn) REFERENCES EMPLOYEE(Ssn),
    ->     FOREIGN KEY (Dno) REFERENCES DEPARTMENT(Dnumber)
    -> );
Query OK, 0 rows affected (0.04 sec)

mysql> ALTER TABLE DEPARTMENT
    -> ADD FOREIGN KEY(Mgr_ssn) REFERENCES EMPLOYEE(Ssn);
Query OK, 0 rows affected (0.07 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

```
mysql> CREATE TABLE DEPARTMENT_LOCATIONS(
    ->     DNumber INT NOT NULL,
    ->     DLocation VARCHAR(15) NOT NULL,
    ->     PRIMARY KEY(DNumber, DLocation),
    ->     FOREIGN KEY(DNumber) REFERENCES DEPARTMENT(Dnumber)
    -> );
Query OK, 0 rows affected (0.02 sec)


mysql> CREATE TABLE PROJECT(
    ->     Pname VARCHAR(15) NOT NULL,
    ->     Pnumber INT NOT NULL,
    ->     Plocation VARCHAR(15),
    ->     Dnum INT NOT NULL,
    ->     PRIMARY KEY(Pnumber),
    ->     UNIQUE(Pname),
    ->     FOREIGN KEY(Dnum) REFERENCES DEPARTMENT(Dnumber)
    -> );
Query OK, 0 rows affected (0.03 sec)


mysql> CREATE TABLE WORKS_ON(
    ->     Essn CHAR(9) NOT NULL,
    ->     Pno INT NOT NULL,
    ->     Hours DECIMAL(3, 1),
    ->     PRIMARY KEY(Essn, Pno),
    ->     FOREIGN KEY(Essn) REFERENCES EMPLOYEE(Ssn),
    ->     FOREIGN KEY(Pno) REFERENCES PROJECT(Pnumber)
    -> );
Query OK, 0 rows affected (0.03 sec)


mysql> CREATE TABLE DEPENDENT(
    ->     Essn CHAR(9) NOT NULL,
    ->     Dependent_name VARCHAR(15) NOT NULL,
    ->     Sex CHAR,
    ->     Bdate DATE,
    ->     Relationship VARCHAR(30),
    ->     PRIMARY KEY(Essn, Dependent_name),
    ->     FOREIGN KEY(Essn) REFERENCES EMPLOYEE(Ssn)
    -> );
Query OK, 0 rows affected (0.02 sec)
```

```
mysql> -- INSERTING DATA
mysql>
mysql> INSERT INTO DEPARTMENT(Dname, Dnumber, Mgr_start_date, Mgr_ssn) VALUES('Research', 5, '1988-05-22', NULL);
Query OK, 1 row affected (0.01 sec)

mysql> INSERT INTO DEPARTMENT(Dname, Dnumber, Mgr_start_date, Mgr_ssn) VALUES('Administration', 4, '1995-01-01', NULL);
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO DEPARTMENT(Dname, Dnumber, Mgr_start_date, Mgr_ssn) VALUES('Headquarters', 1, '1981-06-19', NULL);
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO DEPARTMENT(Dname, Dnumber, Mgr_start_date, Mgr_ssn) VALUES('Software', 7, '1998-06-19', NULL);
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO DEPARTMENT(Dname, Dnumber, Mgr_start_date, Mgr_ssn) VALUES('Services', 3, '1985-01-01', NULL);
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO DEPARTMENT(Dname, Dnumber, Mgr_start_date, Mgr_ssn) VALUES('Finance', 8, '1995-01-01', NULL);
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO DEPARTMENT(Dname, Dnumber, Mgr_start_date, Mgr_ssn) VALUES('Public Relations', 2, '1993-07-08', NULL);
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO DEPARTMENT(Dname, Dnumber, Mgr_start_date, Mgr_ssn) VALUES('Production', 6, '1996-05-01', NULL);
Query OK, 1 row affected (0.00 sec)

mysql>
mysql> INSERT INTO EMPLOYEE(Fname, Minit, Lname, Ssn, Bdate, Address, Sex, Salary, Super_ssn, Dno) VALUES('John', 'B', 'Smith', '12345
6789', '1965-01-09', '731 Fondren, Houston, TX', 'M', 30000.00, NULL, NULL);
Query OK, 1 row affected (0.01 sec)

mysql> INSERT INTO EMPLOYEE(Fname, Minit, Lname, Ssn, Bdate, Address, Sex, Salary, Super_ssn, Dno) VALUES('Franklin', 'T', 'Wong', '33
3445555', '1955-12-08', '638 Voss, Houston, TX', 'M', 40000.00, NULL, NULL);


mysql> INSERT INTO EMPLOYEE(Fname, Minit, Lname, Ssn, Bdate, Address, Sex, Salary, Super_ssn, Dno) VALUES('Alicia', 'J', 'Zelaya', '99
9887777', '1968-01-19', '3321 Castle, Spring, TX', 'F', 25000.00, NULL, NULL);
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO EMPLOYEE(Fname, Minit, Lname, Ssn, Bdate, Address, Sex, Salary, Super_ssn, Dno) VALUES('Jennifer', 'S', 'Wallace',
'987654321', '1941-06-20', '291 Berry, Bellaire, TX', 'F', 43000.00, NULL, NULL);
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO EMPLOYEE(Fname, Minit, Lname, Ssn, Bdate, Address, Sex, Salary, Super_ssn, Dno) VALUES('Ramesh', 'K', 'Narayan', '6
66884444', '1962-09-15', '975 Fire Oak, Humble, TX', 'M', 38000.00, NULL, NULL);
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO EMPLOYEE(Fname, Minit, Lname, Ssn, Bdate, Address, Sex, Salary, Super_ssn, Dno) VALUES('Joyce', 'A', 'English', '45
3453453', '1972-07-31', '5631 Rice, Houston, TX', 'F', 25000.00, NULL, NULL);
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO EMPLOYEE(Fname, Minit, Lname, Ssn, Bdate, Address, Sex, Salary, Super_ssn, Dno) VALUES( 'Ahmad', 'V', 'Jabbar', '88
8665555', '1969-03-29', '980 Dallas, Houston, TX', 'M', 25000.00, NULL, NULL);
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO EMPLOYEE(Fname, Minit, Lname, Ssn, Bdate, Address, Sex, Salary, Super_ssn, Dno) VALUES('James', 'E', 'Borg', '88866
4555', '1937-11-10', '450 Stone, Houston, TX', 'M', 55000.00, NULL, NULL);
Query OK, 1 row affected (0.00 sec)

mysql>
mysql> UPDATE DEPARTMENT SET Mgr_ssn = '123456789' WHERE Dnumber = 5;
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> UPDATE DEPARTMENT SET Mgr_ssn = '333445555' WHERE Dnumber = 4;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

```
mysql> UPDATE DEPARTMENT SET Mgr_ssn = '999887777' WHERE Dnumber = 1;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> UPDATE DEPARTMENT SET Mgr_ssn = '987654321' WHERE Dnumber = 7;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> UPDATE DEPARTMENT SET Mgr_ssn = '666884444' WHERE Dnumber = 3;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> UPDATE DEPARTMENT SET Mgr_ssn = '453453453' WHERE Dnumber = 8;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> UPDATE DEPARTMENT SET Mgr_ssn = '888664555' WHERE Dnumber = 2;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> UPDATE DEPARTMENT SET Mgr_ssn = '888665555' WHERE Dnumber = 6;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql>
mysql> UPDATE EMPLOYEE SET Super_ssn = '987654321', Dno = 7 WHERE Ssn = '123456789';
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> UPDATE EMPLOYEE SET Super_ssn = '987654321', Dno = 7 WHERE Ssn = '333445555';
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0


mysql> UPDATE EMPLOYEE SET Super_ssn = '333445555', Dno = 5 WHERE Ssn = '999887777';
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> UPDATE EMPLOYEE SET Super_ssn = '333445555', Dno = 5 WHERE Ssn = '987654321';
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> UPDATE EMPLOYEE SET Super_ssn = '333445555', Dno = 5 WHERE Ssn = '666884444';
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> UPDATE EMPLOYEE SET Super_ssn = '453453453', Dno = 1 WHERE Ssn = '453453453';
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> UPDATE EMPLOYEE SET Super_ssn = '453453453', Dno = 1 WHERE Ssn = '888665555';
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> UPDATE EMPLOYEE SET Super_ssn = '453453453', Dno = 1 WHERE Ssn = '888664555';
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql>
mysql> INSERT INTO DEPARTMENT_LOCATIONS(DNumber, DLocation) VALUES(1, 'Stafford');
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO DEPARTMENT_LOCATIONS(DNumber, DLocation) VALUES(1, 'Bellaire');
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO DEPARTMENT_LOCATIONS(DNumber, DLocation) VALUES(2, 'Houston');
Query OK, 1 row affected (0.00 sec)
```

```
mysql> INSERT INTO DEPARTMENT_LOCATIONS(DNumber, DLocation) VALUES(2, 'Sugar Land');
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO DEPARTMENT_LOCATIONS(DNumber, DLocation) VALUES(3, 'Bellaire');
Query OK, 1 row affected (0.00 sec)

mysql>
mysql> INSERT INTO PROJECT(Pname, Pnumber, Plocation, Dnum) VALUES('ProductX', 1, 'Bellaire', 5);
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO PROJECT(Pname, Pnumber, Plocation, Dnum) VALUES('ProductY', 2, 'Stafford', 5);
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO PROJECT(Pname, Pnumber, Plocation, Dnum) VALUES('ProductZ', 3, 'Houston', 6);
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO PROJECT(Pname, Pnumber, Plocation, Dnum) VALUES('Computerization', 10, 'Stafford', 1);
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO PROJECT(Pname, Pnumber, Plocation, Dnum) VALUES('Reorganization', 20, 'Houston', 4);
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO PROJECT(Pname, Pnumber, Plocation, Dnum) VALUES('Newbenefits', 30, 'Stafford', 4);
Query OK, 1 row affected (0.00 sec)

mysql>
mysql> INSERT INTO WORKS_ON(Essn, Pno, Hours) VALUES('123456789', 1, 32.5);
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO WORKS_ON(Essn, Pno, Hours) VALUES('333445555', 1, 7.5);
Query OK, 1 row affected (0.00 sec)


mysql> INSERT INTO WORKS_ON(Essn, Pno, Hours) VALUES('999887777', 2, 40.0);
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO WORKS_ON(Essn, Pno, Hours) VALUES('987654321', 3, 20.0);
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO WORKS_ON(Essn, Pno, Hours) VALUES('666884444', 10, 10.0);
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO WORKS_ON(Essn, Pno, Hours) VALUES('453453453', 20, 30.0);
Query OK, 1 row affected (0.00 sec)

mysql>
mysql> INSERT INTO DEPENDENT(Essn, Dependent_name, Sex, Bdate, Relationship) VALUES('123456789', 'John Smith', 'M', '1965-01-09', '731
 Fondren, Houston, TX');
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO DEPENDENT(Essn, Dependent_name, Sex, Bdate, Relationship) VALUES('333445555', 'Franklin Wong', 'M', '1955-12-08', '
638 Voss, Houston, TX');
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO DEPENDENT(Essn, Dependent_name, Sex, Bdate, Relationship) VALUES('999887777', 'Alicia Zelaya', 'F', '1968-01-19', '
3321 Castle, Spring, TX');
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO DEPENDENT(Essn, Dependent_name, Sex, Bdate, Relationship) VALUES('987654321', 'Jennifer Wallace', 'F', '1941-06-20'
, '291 Berry, Bellaire, TX');
ERROR 1406 (22001): Data too long for column 'Dependent_name' at row 1
mysql> INSERT INTO DEPENDENT(Essn,Dependent_name, Sex, Bdate, Relationship) VALUES('666884444', 'Ramesh Narayan', 'M', '1962-09-15', '
975 Fire Oak, Humble, TX');
Query OK, 1 row affected (0.00 sec)


mysql> INSERT INTO DEPENDENT(Essn, Dependent_name, Sex, Bdate, Relationship) VALUES('453453453', 'Joyce English', 'F', '1972-07-31', '
5631 Rice, Houston, TX');
Query OK, 1 row affected (0.00 sec)
```

```
mysql> -- PROBLEM STATEMENT SOLUTIONS
mysql>
mysql> -- 1)   Retrieve the birth date and address of the employee(s) whose name is 'John B. Smith'.
mysql> SELECT Bdate, Address FROM EMPLOYEE
    -> WHERE Fname = 'John' AND Minit = 'B' AND Lname = 'Smith';
+------------+--------------------------+
| Bdate      | Address                  |
+------------+--------------------------+
| 1965-01-09 | 731 Fondren, Houston, TX |
+------------+--------------------------+
1 row in set (0.00 sec)

mysql>
mysql> -- 2)    Retrieve the name and address of all employees who work for the 'Research' department.
mysql> SELECT Fname, Minit, Lname, Address FROM EMPLOYEE,DEPARTMENT
    -> WHERE Dname = 'Research' AND Dnumber = Dno;
+----------+-------+---------+--------------------------+
| Fname    | Minit | Lname   | Address                  |
+----------+-------+---------+--------------------------+
| Ramesh   | K     | Narayan | 975 Fire Oak, Humble, TX |
| Jennifer | S     | Wallace | 291 Berry, Bellaire, TX  |
| Alicia   | J     | Zelaya  | 3321 Castle, Spring, TX  |
+----------+-------+---------+--------------------------+
3 rows in set (0.00 sec)

mysql> -- 3)    For every project located in 'Stafford', list the project number, the controlling department number, and the departmen
t manager's last name, address, and birth date.
mysql> SELECT Lname, address, Bdate FROM EMPLOYEE, PROJECT, DEPARTMENT
    -> WHERE Plocation = 'Stafford' AND Dnumber = Dnum AND Mgr_ssn = Ssn;
+--------+--------------------------+------------+
| Lname  | address                  | Bdate      |
+--------+--------------------------+------------+
| Smith  | 731 Fondren, Houston, TX | 1965-01-09 |
| Zelaya | 3321 Castle, Spring, TX  | 1968-01-19 |
| Wong   | 638 Voss, Houston, TX    | 1955-12-08 |
+--------+--------------------------+------------+
3 rows in set (0.00 sec)
mysql> -- 4)    Select all combinations of EMPLOYEE Ssn and DEPARTMENT Dname in the database.
mysql> SELECT Ssn, Dname
    -> FROM EMPLOYEE, DEPARTMENT;
+-----------+------------------+
| Ssn       | Dname            |
+-----------+------------------+
| 333445555 | Administration   |
| 123456789 | Administration   |
| 999887777 | Administration   |
| 987654321 | Administration   |
| 666884444 | Administration   |
| 888665555 | Administration   |
| 888664555 | Administration   |
| 453453453 | Administration   |
| 333445555 | Finance          |
| 123456789 | Finance          |
| 999887777 | Finance          |
| 987654321 | Finance          |
| 666884444 | Finance          |
| 888665555 | Finance          |
| 888664555 | Finance          |
| 453453453 | Finance          |
| 333445555 | Headquarters     |
| 123456789 | Headquarters     |
| 999887777 | Headquarters     |
| 987654321 | Headquarters     |
| 666884444 | Headquarters     |
| 888665555 | Headquarters     |
| 888664555 | Headquarters     |
| 453453453 | Headquarters     |
| 333445555 | Production       |
| 123456789 | Production       |
```

```
| 999887777 | Production       |
| 987654321 | Production       |
| 666884444 | Production       |
| 888665555 | Production       |
| 888664555 | Production       |
| 453453453 | Production       |
| 333445555 | Public Relations |
| 123456789 | Public Relations |
| 999887777 | Public Relations |
| 987654321 | Public Relations |
| 666884444 | Public Relations |
| 888665555 | Public Relations |
| 888664555 | Public Relations |
| 453453453 | Public Relations |
| 333445555 | Research         |
| 123456789 | Research         |
| 999887777 | Research         |
| 987654321 | Research         |
| 666884444 | Research         |
| 888665555 | Research         |
| 888664555 | Research         |
| 453453453 | Research         |
| 333445555 | Services         |
| 123456789 | Services         |
| 999887777 | Services         |
| 987654321 | Services         |
| 666884444 | Services         |
| 888665555 | Services         |
| 888664555 | Services         |
| 453453453 | Services         |
| 333445555 | Software         |
| 123456789 | Software         |
| 999887777 | Software         |
| 999887777 | Software         |
| 987654321 | Software         |
| 666884444 | Software         |
| 888665555 | Software         |
| 888664555 | Software         |
| 453453453 | Software         |
+-----------+------------------+
64 rows in set (0.00 sec)

mysql> -- 5)    Retrieve all the attribute values of any EMPLOYEE who works in DEPARTMENT number 5.
mysql> SELECT * FROM EMPLOYEE
    -> WHERE Dno = 5;
+----------+-------+---------+-----------+------------+--------------------------+------+----------+-----------+------+
| Fname    | Minit | Lname   | Ssn       | Bdate      | Address                  | Sex  | Salary   | Super_ssn | Dno  |
+----------+-------+---------+-----------+------------+--------------------------+------+----------+-----------+------+
| Ramesh   | K     | Narayan | 666884444 | 1962-09-15 | 975 Fire Oak, Humble, TX | M    | 38000.00 | 333445555 |    5 |
| Jennifer | S     | Wallace | 987654321 | 1941-06-20 | 291 Berry, Bellaire, TX  | F    | 43000.00 | 333445555 |    5 |
| Alicia   | J     | Zelaya  | 999887777 | 1968-01-19 | 3321 Castle, Spring, TX  | F    | 25000.00 | 333445555 |    5 |
+----------+-------+---------+-----------+------------+--------------------------+------+----------+-----------+------+
3 rows in set (0.00 sec)

mysql>
mysql> -- 6)    Retrieve all distinct salary values.
mysql> SELECT DISTINCT Salary
    -> FROM EMPLOYEE;
+----------+
| Salary   |
+----------+
| 30000.00 |
| 40000.00 |
| 25000.00 |
| 38000.00 |
| 55000.00 |
| 43000.00 |
+----------+
6 rows in set (0.00 sec)
```

```
mysql> -- 7)    Make a list of all project numbers for projects that involve an employee whose last name is 'Smith', either as a worke
r or as a manager of the department that controls the project.
mysql> SELECT Pnumber FROM PROJECT, EMPLOYEE
    -> WHERE Lname = 'Smith';
+---------+
| Pnumber |
+---------+
|      10 |
|      20 |
|      30 |
|       1 |
|       2 |
|       3 |
+---------+
6 rows in set (0.00 sec)

mysql>
mysql> -- 8)    Retrieve all employees whose address is in Houston, Texas.
mysql> SELECT * FROM EMPLOYEE
    -> WHERE Address LIKE '%Houston, TX';
+----------+-------+---------+-----------+------------+-------------------------+------+----------+-----------+------+
| Fname    | Minit | Lname   | Ssn       | Bdate      | Address                 | Sex  | Salary   | Super_ssn | Dno  |
+----------+-------+---------+-----------+------------+-------------------------+------+----------+-----------+------+
| John     | B     | Smith   | 123456789 | 1965-01-09 | 731 Fondren, Houston, TX | M    | 30000.00 | 987654321 |    7 |
| Franklin | T     | Wong    | 333445555 | 1955-12-08 | 638 Voss, Houston, TX    | M    | 40000.00 | 987654321 |    7 |
| Joyce    | A     | English | 453453453 | 1972-07-31 | 5631 Rice, Houston, TX   | F    | 25000.00 | 453453453 |    1 |
| James    | E     | Borg    | 888664555 | 1937-11-10 | 450 Stone, Houston, TX   | M    | 55000.00 | 453453453 |    1 |
| Ahmad    | V     | Jabbar  | 888665555 | 1969-03-29 | 980 Dallas, Houston, TX  | M    | 25000.00 | 453453453 |    1 |
+----------+-------+---------+-----------+------------+-------------------------+------+----------+-----------+------+
5 rows in set (0.00 sec)
mysql> -- 9)    Find all employees who were born during the 1950s
mysql> SELECT * FROM EMPLOYEE WHERE Bdate BETWEEN '1950-01-01' AND '1959-12-31';
+----------+-------+-------+-----------+------------+-----------------------+------+----------+-----------+------+
| Fname    | Minit | Lname | Ssn       | Bdate      | Address               | Sex  | Salary   | Super_ssn | Dno  |
+----------+-------+-------+-----------+------------+-----------------------+------+----------+-----------+------+
| Franklin | T     | Wong  | 333445555 | 1955-12-08 | 638 Voss, Houston, TX | M    | 40000.00 | 987654321 |    7 |
+----------+-------+-------+-----------+------------+-----------------------+------+----------+-----------+------+
1 row in set (0.00 sec)

mysql>
mysql> -- 10)   Show the resulting salaries if every employee working on the 'ProductX' project is given a 10 percent raise.
mysql> SELECT Salary as Previous_salary, Salary + Salary * 0.1 Raised_salary FROM EMPLOYEE, WORKS_ON
    -> WHERE Pno = 1;
+-----------------+----------------+
| Previous_salary | Raised_salary  |
+-----------------+----------------+
|        30000.00 |     33000.000 |
|        30000.00 |     33000.000 |
|        40000.00 |     44000.000 |
|        40000.00 |     44000.000 |
|        25000.00 |     27500.000 |
|        25000.00 |     27500.000 |
|        38000.00 |     41800.000 |
|        38000.00 |     41800.000 |
|        55000.00 |     60500.000 |
|        55000.00 |     60500.000 |
|        25000.00 |     27500.000 |
|        25000.00 |     27500.000 |
|        43000.00 |     47300.000 |
|        43000.00 |     47300.000 |
|        25000.00 |     27500.000 |
|        25000.00 |     27500.000 |
+-----------------+----------------+
mysql> -- 11)   Retrieve a list of employees and the projects they are working on, ordered by department and, within each department,
ordered alphabetically by last name, then first name.
mysql> SELECT Fname, Lname, Pname FROM EMPLOYEE, PROJECT, WORKS_ON
    -> WHERE EMPLOYEE.Ssn = WORKS_ON.Essn AND PROJECT.Pnumber = WORKS_ON.Pno
    -> ORDER BY Dno, Lname, Fname;
+----------+---------+-----------------+
| Fname    | Lname   | Pname           |
+----------+---------+-----------------+
| Joyce    | English | Reorganization  |
| Ramesh   | Narayan | Computerization |
| Jennifer | Wallace | ProductZ        |
| Alicia   | Zelaya  | ProductY        |
| John     | Smith   | ProductX        |
| Franklin | Wong    | ProductX        |
+----------+---------+-----------------+
6 rows in set (0.00 sec)
```

Preparatory Questions

Q1) Which operator performs Pattern matching?

1) BETWEEN operators
2) LIKE operator
3) EXISTS operator.
4) None of the above

Q2) In SQL which commands are used to change the storage characteristics of the table?

1) ALTER TABLE
2) MODIFY TABLE
3) CHANGE TABLE
4) All of the above

Q3) _____removes all rows from a table without logging the individual row deletions.

1) DELETE
2) REMOVE
3) DROP
4) TRUNCATE

Q4) If you don't specify ASC or DESC after a SQL ORDER BY clause, the following is used by default :

1) ASC
2) DESC
3) There is no default value
4) None of the mentioned

Q5) What is the purpose of the SQL AS clause?

1) The AS SQL clause is used to change the name of a column in the result set or to assign a name to a derived column
2) The AS clause is used with the JOIN clause only
3) The AS clause defines a search condition
4) All of the mentioned

Experiment No: 5

Student Name and Roll Number: Piyush Gambhir – 21CSU349
Semester /Section: Semester-IV – AIML-B (AL-3)
Link to Code: Piyush-Gambhir/NCU-CSL214-DBMS-Lab_Manual (github.com)
Date: 27.02.2023
Faculty Signature:
Marks:

Objective

To familiarize with JOIN operations in SQL on the COMPANY database.

Program Outcome

- The students will be to establish a connection between two or more database tables based on matching columns, thereby creating a relationship between the tables.

Problem Statement

Consider the Sample database given below and answer the queries as stated:

Pack_grades

Grade_id	Grade_name	Min_price	Max_price

Customers

Customer_id	First_name	Last_name	Birth_date	Join_date	City	Pack_id	State

Packages

Pack_id	Speed	Start_date	Monthly_payment	Sector_id

Sectors

Sector_id	Sector_name

1) Write a query to display first name, last name, package number and internet speed for all customers.
2) Write a query to display first name, last name, package number and internet speed for all customers whose package number equals 22 or 27. Order the query in ascending order by last name.
3) Display the package number, internet speed, monthly payment and sector name for all packages (*Packages* and *Sectors* tables).
4) Display the customer name, package number, internet speed, monthly payment and sector name for all customers (*Customers*, *Packages* and *Sectors* tables).
5) Display the customer name, package number, internet speed, monthly payment and sector name for all customers in the business sector (*Customers*, *Packages* and *Sectors* tables).
6) Display the last name, first name, join date, package number, internet speed and sector name for all customers in the private sector who joined the company in the year 2006.
7) Display the package number, internet speed, monthly payment and package grade for all packages (*Packages* and *Pack_Grades* tables).
8) Display the first name, last name, internet speed and monthly payment for all customers. Use INNER JOIN to solve this exercise.
9) Display the last name, first name and package number for all customers who have the same package number as customer named 'Amado Taylor' (*Customers* table).
10) Display the package number and internet speed for all packages whose internet speed is equal to the internet speed of package number 10 (*Packages* table).

Background Study

1) JOINED TABLE: Permits users to specify a table resulting from a join operation in the FROM clause of a query
   - The attributes of such a table are all the attributes of the first table followed by all the attributes of the second table.
   - The default type of join in a joined table is called an inner join, where a tuple is included in the result only if a matching tuple exists in the other relation.

2) NATURAL JOIN on two relations R and S
   - No join condition specified.
   - Implicit EQUIJOIN condition for each pair of attributes with same name from R & S

- It is possible to rename the attributes so that they match, if the names of the join attributes are not the same in the base relations.

3) LEFT OUTER JOIN
   - Every tuple in left table must appear in result
   - If no matching tuple, padded with NULL values for attributes of right table

4) RIGHT OUTER JOIN
   - Every tuple in right table must appear in result
   - If no matching tuple, padded with NULL values for the attributes of left table

5) FULL OUTER JOIN
   - Used when both the tables are taken completely

6) CROSS JOIN – for Cartesian Product

Output: Screenshots

```
mysql> -- DROP DATABASE IF EXISTS EXPERIMENT_5;
mysql> DROP DATABASE IF EXISTS EXPERIMENT_5;
Query OK, 4 rows affected (0.05 sec)

mysql>
mysql> -- CREATING DATABASE
mysql> CREATE DATABASE EXPERIMENT_5;
Query OK, 1 row affected (0.00 sec)

mysql>
mysql> -- USING DATABASE
mysql> USE EXPERIMENT_5;
Database changed
mysql> -- CREATING TABLES
mysql> CREATE TABLE Pack_grades(
    ->     Grade_id INT NOT NULL,
    ->     Grade_name VARCHAR(20) NOT NULL,
    ->     Min_price INT NOT NULL,
    ->     Max_price INT NOT NULL,
    ->     PRIMARY KEY(Grade_id)
    -> );
Query OK, 0 rows affected (0.01 sec)

mysql>
mysql> CREATE TABLE Customers(
    ->     Customer_id INT NOT NULL,
    ->     First_name VARCHAR(20) NOT NULL,
    ->     Last_name VARCHAR(20) NOT NULL,
    ->     Birth_date DATE NOT NULL,
    ->     Join_date DATE NOT NULL,
    ->     City VARCHAR(20) NOT NULL,
    ->     Pack_id INT,
    ->     State VARCHAR(20) NOT NULL,
    ->     PRIMARY KEY(Customer_id)
    -> );
Query OK, 0 rows affected (0.01 sec)

mysql> CREATE TABLE Packages(
    ->     Pack_id INT NOT NULL,
    ->     Speed INT NOT NULL,
    ->     Start_date DATE NOT NULL,
    ->     Monthly_payment INT NOT NULL,
    ->     Sector_id INT,
    ->     PRIMARY KEY(Pack_id)
    -> );
Query OK, 0 rows affected (0.01 sec)

mysql>
mysql> CREATE TABLE Sectors(
    ->     Sector_id INT NOT NULL,
    ->     Sector_name VARCHAR(20) NOT NULL,
    ->     PRIMARY KEY(Sector_id)
    -> );
Query OK, 0 rows affected (0.01 sec)

mysql>
mysql> ALTER TABLE Customers ADD
    -> FOREIGN KEY(Pack_id) REFERENCES Packages(Pack_id);
Query OK, 0 rows affected (0.05 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql>
mysql> ALTER TABLE Packages ADD
    -> FOREIGN KEY(Sector_id) REFERENCES Sectors(Sector_id);
Query OK, 0 rows affected (0.04 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

```
mysql> -- INSERTING VALUES INTO TABLES
mysql>
mysql> INSERT INTO Pack_grades(Grade_id, Grade_name, Min_price, Max_price) VALUES(1, 'A', 0, 100);
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO Pack_grades(Grade_id, Grade_name, Min_price, Max_price) VALUES(2, 'B', 101, 200);
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO Pack_grades(Grade_id, Grade_name, Min_price, Max_price) VALUES(3, 'C', 201, 300);
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO Pack_grades(Grade_id, Grade_name, Min_price, Max_price) VALUES(4, 'D', 301, 400);
Query OK, 1 row affected (0.00 sec)

mysql>
mysql> INSERT INTO Customers(Customer_id, First_name, Last_name, Birth_date, Join_date, City, Pack_id, State) VALUES(1, 'Amado', 'Tayl
or', '1990-01-01', '2006-01-01', 'New York', NULL, 'NY');
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO Customers(Customer_id, First_name, Last_name, Birth_date, Join_date, City, Pack_id, State) VALUES(2, 'Luis', 'Gonza
lez', '1990-01-01', '2006-01-01', 'New York', NULL, 'NY');
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO Customers(Customer_id, First_name, Last_name, Birth_date, Join_date, City, Pack_id, State) VALUES(3, 'Maria', 'Gonz
alez', '1990-01-01', '2006-01-01', 'New York', NULL, 'NY');
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO Customers(Customer_id, First_name, Last_name, Birth_date, Join_date, City, Pack_id, State) VALUES(4, 'Jose', 'Gonza
lez', '1990-01-01', '2006-01-01', 'New York', NULL, 'NY');
Query OK, 1 row affected (0.00 sec)


mysql> INSERT INTO Packages(Pack_id, Speed, Start_date, Monthly_payment, Sector_id) VALUES(22, 10, '2006-01-01', 100, NULL);
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO Packages(Pack_id, Speed, Start_date, Monthly_payment, Sector_id) VALUES(27, 20, '2006-01-01', 200, NULL);
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO Packages(Pack_id, Speed, Start_date, Monthly_payment, Sector_id) VALUES(28, 30, '2006-01-01', 300, NULL);
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO Packages(Pack_id, Speed, Start_date, Monthly_payment, Sector_id) VALUES(29, 40, '2006-01-01', 400, NULL);
Query OK, 1 row affected (0.00 sec)

mysql>
mysql> INSERT INTO Sectors(Sector_id, Sector_name) VALUES(1, 'Private');
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO Sectors(Sector_id, Sector_name) VALUES(2, 'Business');
Query OK, 1 row affected (0.00 sec)

mysql>
mysql> UPDATE Customers SET Pack_id = 27 WHERE Customer_id = 1;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> UPDATE Customers SET Pack_id = 22 WHERE Customer_id = 2;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> UPDATE Customers SET Pack_id = 29 WHERE Customer_id = 3;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

```
mysql> UPDATE Customers SET Pack_id = 28 WHERE Customer_id = 4;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql>
mysql> UPDATE Packages SET Sector_id = 1 WHERE Pack_id = 22;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> UPDATE Packages SET Sector_id = 1 WHERE Pack_id = 27;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> UPDATE Packages SET Sector_id = 2 WHERE Pack_id = 29;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> UPDATE Packages SET Sector_id = 2 WHERE Pack_id = 28;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0


mysql> -- PROBLEM STATEMENT SOLUTIONS
mysql>
mysql> -- 1)    Write a query to display first name, last name, package number and internet speed for all customers.
mysql> SELECT Customers.First_name, Customers.Last_name, Packages.Pack_id, Packages.Speed FROM Customers
    -> INNER JOIN Packages ON Customers.Pack_id = Packages.Pack_id;
+------------+-----------+---------+-------+
| First_name | Last_name | Pack_id | Speed |
+------------+-----------+---------+-------+
| Amado      | Taylor    |      27 |    20 |
| Luis       | Gonzalez  |      22 |    10 |
| Maria      | Gonzalez  |      29 |    40 |
| Jose       | Gonzalez  |      28 |    30 |
+------------+-----------+---------+-------+
4 rows in set (0.00 sec)

mysql>
mysql> -- 2)    Write a query to display first name, last name, package number and internet speed for all customers whose package numb
er equals 22 or 27. Order the query in ascending order by last name.
mysql> SELECT Customers.First_name, Customers.Last_name, Packages.Pack_id, Packages.Speed FROM Customers
    -> INNER JOIN Packages ON Customers.Pack_id = Packages.Pack_id
    -> WHERE Packages.Pack_id = 22 OR Packages.Pack_id = 27
    -> ORDER BY Customers.Last_name ASC;
+------------+-----------+---------+-------+
| First_name | Last_name | Pack_id | Speed |
+------------+-----------+---------+-------+
| Luis       | Gonzalez  |      22 |    10 |
| Amado      | Taylor    |      27 |    20 |
+------------+-----------+---------+-------+
2 rows in set (0.00 sec)

mysql> -- 3)    Display the package number, internet speed, monthly payment and sector name for all packages (Packages and Sectors tab
les).
mysql> SELECT Packages.Pack_id, Packages.Speed, Packages.Monthly_payment, Sectors.Sector_name FROM Packages
    -> INNER JOIN Sectors ON Packages.Sector_id = Sectors.Sector_id;
+---------+-------+-----------------+-------------+
| Pack_id | Speed | Monthly_payment | Sector_name |
+---------+-------+-----------------+-------------+
|      22 |    10 |             100 | Private     |
|      27 |    20 |             200 | Private     |
|      28 |    30 |             300 | Business    |
|      29 |    40 |             400 | Business    |
+---------+-------+-----------------+-------------+
4 rows in set (0.00 sec)

mysql>
mysql> -- 4)    Display the customer name, package number, internet speed, monthly payment and sector name for all customers (Customer
s, Packages and Sectors tables).
mysql> SELECT Customers.First_name, Customers.Last_name, Packages.Pack_id, Packages.Speed, Packages.Monthly_payment, Sectors.Sector_na
me FROM Customers
    -> INNER JOIN Packages ON Customers.Pack_id = Packages.Pack_id
    -> INNER JOIN Sectors ON Packages.Sector_id = Sectors.Sector_id;
+------------+-----------+---------+-------+-----------------+-------------+
| First_name | Last_name | Pack_id | Speed | Monthly_payment | Sector_name |
+------------+-----------+---------+-------+-----------------+-------------+
| Luis       | Gonzalez  |      22 |    10 |             100 | Private     |
| Amado      | Taylor    |      27 |    20 |             200 | Private     |
| Jose       | Gonzalez  |      28 |    30 |             300 | Business    |
| Maria      | Gonzalez  |      29 |    40 |             400 | Business    |
+------------+-----------+---------+-------+-----------------+-------------+
4 rows in set (0.00 sec)
```

```
mysql> -- 5)    Display the customer name, package number, internet speed, monthly payment and sector name for all customers in the bu
siness sector (Customers, Packages and Sectors tables).
mysql> SELECT Customers.First_name, Customers.Last_name, Packages.Pack_id, Packages.Speed, Packages.Monthly_payment, Sectors.Sector_na
me FROM Customers
    -> INNER JOIN Packages ON Customers.Pack_id = Packages.Pack_id
    -> INNER JOIN Sectors ON Packages.Sector_id = Sectors.Sector_id
    -> WHERE Sectors.Sector_name = 'Business';
+------------+-----------+---------+-------+-----------------+-------------+
| First_name | Last_name | Pack_id | Speed | Monthly_payment | Sector_name |
+------------+-----------+---------+-------+-----------------+-------------+
| Jose       | Gonzalez  |      28 |    30 |             300 | Business    |
| Maria      | Gonzalez  |      29 |    40 |             400 | Business    |
+------------+-----------+---------+-------+-----------------+-------------+
2 rows in set (0.00 sec)

mysql>
mysql> -- 6)    Display the last name, first name, join date, package number, internet speed and sector name for all customers in the
private sector who joined the company in the year 2006.
mysql> SELECT Customers.Last_name, Customers.First_name, Customers.Join_date, Packages.Pack_id, Packages.Speed, Sectors.Sector_name FR
OM Customers
    -> INNER JOIN Packages ON Customers.Pack_id = Packages.Pack_id
    -> INNER JOIN Sectors ON Packages.Sector_id = Sectors.Sector_id
    -> WHERE Sectors.Sector_name = 'Private' AND Customers.Join_date LIKE '2006%';
+-----------+------------+------------+---------+-------+-------------+
| Last_name | First_name | Join_date  | Pack_id | Speed | Sector_name |
+-----------+------------+------------+---------+-------+-------------+
| Gonzalez  | Luis       | 2006-01-01 |      22 |    10 | Private     |
| Taylor    | Amado      | 2006-01-01 |      27 |    20 | Private     |
+-----------+------------+------------+---------+-------+-------------+
2 rows in set (0.00 sec)

mysql> -- 7)    Display the package number, internet speed, monthly payment and Customer_id for all packages (Packages and Customers t
able).
mysql> SELECT Packages.Pack_id, Packages.Speed, Packages.Monthly_payment, Customers.Customer_id FROM Packages
    -> INNER JOIN Customers ON Packages.Pack_id = Customers.Pack_id;
+---------+-------+-----------------+-------------+
| Pack_id | Speed | Monthly_payment | Customer_id |
+---------+-------+-----------------+-------------+
|      22 |    10 |             100 |           2 |
|      27 |    20 |             200 |           1 |
|      28 |    30 |             300 |           4 |
|      29 |    40 |             400 |           3 |
+---------+-------+-----------------+-------------+
4 rows in set (0.00 sec)

mysql>
mysql> -- 8)    Display the first name, last name, internet speed and monthly payment for all customers. Use INNER JOIN to solve this
exercise.
mysql> SELECT Customers.First_name, Customers.Last_name, Packages.Speed, Packages.Monthly_payment FROM Customers
    -> INNER JOIN Packages ON Customers.Pack_id = Packages.Pack_id;
+------------+-----------+-------+-----------------+
| First_name | Last_name | Speed | Monthly_payment |
+------------+-----------+-------+-----------------+
| Amado      | Taylor    |    20 |             200 |
| Luis       | Gonzalez  |    10 |             100 |
| Maria      | Gonzalez  |    40 |             400 |
| Jose       | Gonzalez  |    30 |             300 |
+------------+-----------+-------+-----------------+
4 rows in set (0.00 sec)

mysql> -- 9)     Display the last name, first name and package number for all customers who have the same package number as customer n
amed ‘Amado Taylor’ (Customers table).
mysql> SELECT Customers.Last_name, Customers.First_name, Customers.Pack_id FROM Customers
    -> WHERE Customers.Pack_id = (SELECT Customers.Pack_id FROM Customers WHERE Customers.First_name = 'Amado' AND Customers.Last_name
 = 'Taylor');
+-----------+------------+---------+
| Last_name | First_name | Pack_id |
+-----------+------------+---------+
| Taylor    | Amado      |      27 |
+-----------+------------+---------+
1 row in set (0.00 sec)

mysql>
mysql> -- 10)    Display the package number and internet speed for all packages whose internet speed is equal to the internet speed of
 package number 10 (Packages table).
mysql> SELECT Packages.Pack_id, Packages.Speed FROM Packages
    -> WHERE Packages.Speed = (SELECT Packages.Speed FROM Packages WHERE Packages.Pack_id = 10);
Empty set (0.00 sec)
```

Preparatory Questions

Q1) Database table by name Loan_Records is given below.

Borrower	Bank_Manager	Loan_Amount
Ramesh	Sunderajan	10000.00
Suresh	Ramgopal	5000.00
Mahesh	Sunderajan	7000.00

What is the output of the following SQL query?

```
SELECT Count(*)
FROM  ( ( SELECT Borrower, Bank_Manager
        FROM Loan_Records) AS S
        NATURAL JOIN ( SELECT Bank_Manager, Loan_Amount
                FROM Loan_Records) AS T );
```

1) 3
2) 9
3) 5
4) 6

Q2) Which product is returned in a join query have no join condition:
1) Equijoins
2) Cartesian
3) Both Equijoins and Cartesian
4) None of the mentioned

Q3) Which join refers to join records from the write table that have no matching key in the left table are include in the result set:
1) Left outer join
2) Right outer join
3) Full outer join
4) Half outer join

Q4) Which operation are allowed in a join view:
1) UPDATE

2) INSERT
3) DELETE
4) All of the mentioned

Q5) Which view that contains more than one table in the top-level FROM clause of the SELECT statement:
1) Join view
2) Datable join view
3) Updatable join view
4) All of the mentioned

Experiment No: 6

Student Name and Roll Number: Piyush Gambhir – 21CSU349
Semester /Section: Semester-IV – AIML-B (AL-3)
Link to Code: Piyush-Gambhir/NCU-CSL214-DBMS-Lab_Manual (github.com)
Date: 13.03.2023
Faculty Signature:
Marks:

Objective

To understand Aggregate functions using SQL queries on the COMPANY database.

Program Outcome

- The students will understand the need of applying an aggregate function to get a single value from a set of values, thus, expressing the significance of the data it is computed from.

Problem Statement

Consider the COMPANY database in Experiment 1 and execute the following queries:

1) Find the sum of the salaries of all employees, the maximum salary, the minimum salary, and the average salary.
2) Find the sum of the salaries of all employees of the ‘Research’ department, as well as the maximum salary, the minimum salary, and the average salary in this department.
3) Retrieve the total number of employees in the company.
4) Retrieve the number of employees in the ‘Research’ department.
5) Count the number of distinct salary values in the database.
6) For each department, retrieve the department number, the number of employees in the department, and their average salary.
7) For each project, retrieve the project number, the project name, and the number of employees who work on that project.

8) For each project *on which more than two employees work*, retrieve the project number, the project name, and the number of employees who work on the project
9) For each project, retrieve the project number, the project name, and the number of employees from department 5 who work on the project.
10) For each department that has more than five employees, retrieve the department number and the number of its employees who are making more than $40,000.

Background Study

1) Aggregate Functions: Used to summarize information from multiple tuples into a single-tuple summary

2) Grouping: Create subgroups of tuples before summarizing

3) Built-in aggregate functions: COUNT, SUM, MAX, MIN, and AVG (NULL values discarded when aggregate functions are applied to a particular column)

4) Functions can be used in the SELECT clause or in a HAVING clause

5) Partition relation into subsets of tuples
   - Based on grouping attribute(s)
   - Apply function to each such group independently

6) GROUP BY clause: Specifies grouping attributes

7) If NULLs exist in grouping attribute, then separate group created for all tuples with a NULL value in grouping attribute

8) HAVING clause provides a condition on the summary information

9) SELECT clause includes only the grouping attribute and the aggregate functions to be applied on each group of tuples.

10) WHERE clause limit the *tuples* to which functions are applied, the HAVING clause serves to choose *whole groups*

Output: Screenshots

```
mysql> -- DROP DATABASE IF EXISTS EXPERIMENT_6;
mysql> DROP DATABASE IF EXISTS EXPERIMENT_6;
Query OK, 0 rows affected, 1 warning (0.01 sec)

mysql>
mysql> -- CREATING DATABASE
mysql> CREATE DATABASE EXPERIMENT_6;
Query OK, 1 row affected (0.01 sec)

mysql>
mysql> -- USING DATABASE
mysql> USE EXPERIMENT_6;
Database changed
mysql> -- CREATING TABLES
mysql> CREATE TABLE DEPARTMENT(
    ->     Dname VARCHAR(30) NOT NULL,
    ->     Dnumber INT NOT NULL,
    ->     Mgr_ssn CHAR(9),
    ->     Mgr_start_date DATE,
    ->     PRIMARY KEY(Dnumber),
    ->     UNIQUE(Dname)
    -> );
Query OK, 0 rows affected (0.04 sec)

mysql>
mysql>
mysql> CREATE TABLE EMPLOYEE(
    ->     Fname VARCHAR(15) NOT NULL,
    ->     Minit CHAR,
    ->     Lname VARCHAR(15) NOT NULL,
    ->     Ssn CHAR(9) NOT NULL,
    ->     Bdate DATE,
    ->     Address VARCHAR(30),
    ->     Sex CHAR,
    ->     Salary DECIMAL(10, 2),
    ->     Super_ssn CHAR(9),
    ->     Dno INT,
    ->     PRIMARY KEY (Ssn),
    ->     FOREIGN KEY (Super_ssn) REFERENCES EMPLOYEE(Ssn),
    ->     FOREIGN KEY (Dno) REFERENCES DEPARTMENT(Dnumber)
    -> );
Query OK, 0 rows affected (0.03 sec)
mysql> ALTER TABLE DEPARTMENT
    -> ADD FOREIGN KEY(Mgr_ssn) REFERENCES EMPLOYEE(Ssn);
Query OK, 0 rows affected (0.05 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql>
mysql> CREATE TABLE DEPARTMENT_LOCATIONS(
    ->     DNumber INT NOT NULL,
    ->     DLocation VARCHAR(15) NOT NULL,
    ->     PRIMARY KEY(DNumber, DLocation),
    ->     FOREIGN KEY(DNumber) REFERENCES DEPARTMENT(Dnumber)
    -> );
Query OK, 0 rows affected (0.01 sec)

mysql>
mysql> CREATE TABLE PROJECT(
    ->     Pname VARCHAR(15) NOT NULL,
    ->     Pnumber INT NOT NULL,
    ->     Plocation VARCHAR(15),
    ->     Dnum INT NOT NULL,
    ->     PRIMARY KEY(Pnumber),
    ->     UNIQUE(Pname),
    ->     FOREIGN KEY(Dnum) REFERENCES DEPARTMENT(Dnumber)
    -> );
Query OK, 0 rows affected (0.03 sec)
```

```
mysql> CREATE TABLE WORKS_ON(
    ->     Essn CHAR(9) NOT NULL,
    ->     Pno INT NOT NULL,
    ->     Hours DECIMAL(3, 1),
    ->     PRIMARY KEY(Essn, Pno),
    ->     FOREIGN KEY(Essn) REFERENCES EMPLOYEE(Ssn),
    ->     FOREIGN KEY(Pno) REFERENCES PROJECT(Pnumber)
    -> );
Query OK, 0 rows affected (0.03 sec)

mysql>
mysql> CREATE TABLE DEPENDENT(
    ->     Essn CHAR(9) NOT NULL,
    ->     Dependent_name VARCHAR(15) NOT NULL,
    ->     Sex CHAR,
    ->     Bdate DATE,
    ->     Relationship VARCHAR(30),
    ->     PRIMARY KEY(Essn, Dependent_name),
    ->     FOREIGN KEY(Essn) REFERENCES EMPLOYEE(Ssn)
    -> );
Query OK, 0 rows affected (0.01 sec)


mysql> -- INSERTING DATA
mysql>
mysql> INSERT INTO DEPARTMENT(Dname, Dnumber, Mgr_start_date, Mgr_ssn) VALUES('Research', 5, '1988-05-22', NULL);
Query OK, 1 row affected (0.01 sec)

mysql> INSERT INTO DEPARTMENT(Dname, Dnumber, Mgr_start_date, Mgr_ssn) VALUES('Administration', 4, '1995-01-01', NULL);
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO DEPARTMENT(Dname, Dnumber, Mgr_start_date, Mgr_ssn) VALUES('Headquarters', 1, '1981-06-19', NULL);
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO DEPARTMENT(Dname, Dnumber, Mgr_start_date, Mgr_ssn) VALUES('Software', 7, '1998-06-19', NULL);
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO DEPARTMENT(Dname, Dnumber, Mgr_start_date, Mgr_ssn) VALUES('Services', 3, '1985-01-01', NULL);
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO DEPARTMENT(Dname, Dnumber, Mgr_start_date, Mgr_ssn) VALUES('Finance', 8, '1995-01-01', NULL);
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO DEPARTMENT(Dname, Dnumber, Mgr_start_date, Mgr_ssn) VALUES('Public Relations', 2, '1993-07-08', NULL);
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO DEPARTMENT(Dname, Dnumber, Mgr_start_date, Mgr_ssn) VALUES('Production', 6, '1996-05-01', NULL);
Query OK, 1 row affected (0.00 sec)

mysql>
mysql> INSERT INTO EMPLOYEE(Fname, Minit, Lname, Ssn, Bdate, Address, Sex, Salary, Super_ssn, Dno) VALUES('John', 'B', 'Smith', '12345
6789', '1965-01-09', '731 Fondren, Houston, TX', 'M', 30000.00, NULL, NULL);
Query OK, 1 row affected (0.01 sec)

mysql> INSERT INTO EMPLOYEE(Fname, Minit, Lname, Ssn, Bdate, Address, Sex, Salary, Super_ssn, Dno) VALUES('Franklin', 'T', 'Wong', '33
3445555', '1955-12-08', '638 Voss, Houston, TX', 'M', 40000.00, NULL, NULL);
```

```
mysql> INSERT INTO EMPLOYEE(Fname, Minit, Lname, Ssn, Bdate, Address, Sex, Salary, Super_ssn, Dno) VALUES('Alicia', 'J', 'Zelaya', '99
9887777', '1968-01-19', '3321 Castle, Spring, TX', 'F', 25000.00, NULL, NULL);
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO EMPLOYEE(Fname, Minit, Lname, Ssn, Bdate, Address, Sex, Salary, Super_ssn, Dno) VALUES('Jennifer', 'S', 'Wallace',
'987654321', '1941-06-20', '291 Berry, Bellaire, TX', 'F', 43000.00, NULL, NULL);
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO EMPLOYEE(Fname, Minit, Lname, Ssn, Bdate, Address, Sex, Salary, Super_ssn, Dno) VALUES('Ramesh', 'K', 'Narayan', '6
66884444', '1962-09-15', '975 Fire Oak, Humble, TX', 'M', 38000.00, NULL, NULL);
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO EMPLOYEE(Fname, Minit, Lname, Ssn, Bdate, Address, Sex, Salary, Super_ssn, Dno) VALUES('Joyce', 'A', 'English', '45
3453453', '1972-07-31', '5631 Rice, Houston, TX', 'F', 25000.00, NULL, NULL);
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO EMPLOYEE(Fname, Minit, Lname, Ssn, Bdate, Address, Sex, Salary, Super_ssn, Dno) VALUES( 'Ahmad', 'V', 'Jabbar', '88
8665555', '1969-03-29', '980 Dallas, Houston, TX', 'M', 25000.00, NULL, NULL);
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO EMPLOYEE(Fname, Minit, Lname, Ssn, Bdate, Address, Sex, Salary, Super_ssn, Dno) VALUES('James', 'E', 'Borg', '88866
4555', '1937-11-10', '450 Stone, Houston, TX', 'M', 55000.00, NULL, NULL);
Query OK, 1 row affected (0.00 sec)

mysql>
mysql> UPDATE DEPARTMENT SET Mgr_ssn = '123456789' WHERE Dnumber = 5;
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> UPDATE DEPARTMENT SET Mgr_ssn = '333445555' WHERE Dnumber = 4;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> UPDATE DEPARTMENT SET Mgr_ssn = '999887777' WHERE Dnumber = 1;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> UPDATE DEPARTMENT SET Mgr_ssn = '987654321' WHERE Dnumber = 7;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> UPDATE DEPARTMENT SET Mgr_ssn = '666884444' WHERE Dnumber = 3;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> UPDATE DEPARTMENT SET Mgr_ssn = '453453453' WHERE Dnumber = 8;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> UPDATE DEPARTMENT SET Mgr_ssn = '888664555' WHERE Dnumber = 2;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> UPDATE DEPARTMENT SET Mgr_ssn = '888665555' WHERE Dnumber = 6;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql>
mysql> UPDATE EMPLOYEE SET Super_ssn = '987654321', Dno = 7 WHERE Ssn = '123456789';
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> UPDATE EMPLOYEE SET Super_ssn = '987654321', Dno = 7 WHERE Ssn = '333445555';
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

```
mysql> UPDATE EMPLOYEE SET Super_ssn = '333445555', Dno = 5 WHERE Ssn = '999887777';
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> UPDATE EMPLOYEE SET Super_ssn = '333445555', Dno = 5 WHERE Ssn = '987654321';
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> UPDATE EMPLOYEE SET Super_ssn = '333445555', Dno = 5 WHERE Ssn = '666884444';
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> UPDATE EMPLOYEE SET Super_ssn = '453453453', Dno = 1 WHERE Ssn = '453453453';
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> UPDATE EMPLOYEE SET Super_ssn = '453453453', Dno = 1 WHERE Ssn = '888665555';
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> UPDATE EMPLOYEE SET Super_ssn = '453453453', Dno = 1 WHERE Ssn = '888664555';
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql>
mysql> INSERT INTO DEPARTMENT_LOCATIONS(DNumber, DLocation) VALUES(1, 'Stafford');
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO DEPARTMENT_LOCATIONS(DNumber, DLocation) VALUES(1, 'Bellaire');
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO DEPARTMENT_LOCATIONS(DNumber, DLocation) VALUES(2, 'Houston');
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO DEPARTMENT_LOCATIONS(DNumber, DLocation) VALUES(2, 'Sugar Land');
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO DEPARTMENT_LOCATIONS(DNumber, DLocation) VALUES(3, 'Bellaire');
Query OK, 1 row affected (0.00 sec)

mysql>
mysql> INSERT INTO PROJECT(Pname, Pnumber, Plocation, Dnum) VALUES('ProductX', 1, 'Bellaire', 5);
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO PROJECT(Pname, Pnumber, Plocation, Dnum) VALUES('ProductY', 2, 'Stafford', 5);
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO PROJECT(Pname, Pnumber, Plocation, Dnum) VALUES('ProductZ', 3, 'Houston', 6);
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO PROJECT(Pname, Pnumber, Plocation, Dnum) VALUES('Computerization', 10, 'Stafford', 1);
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO PROJECT(Pname, Pnumber, Plocation, Dnum) VALUES('Reorganization', 20, 'Houston', 4);
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO PROJECT(Pname, Pnumber, Plocation, Dnum) VALUES('Newbenefits', 30, 'Stafford', 4);
Query OK, 1 row affected (0.00 sec)

mysql>
mysql> INSERT INTO WORKS_ON(Essn, Pno, Hours) VALUES('123456789', 1, 32.5);
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO WORKS_ON(Essn, Pno, Hours) VALUES('333445555', 1, 7.5);
Query OK, 1 row affected (0.00 sec)
```

```
mysql> INSERT INTO WORKS_ON(Essn, Pno, Hours) VALUES('999887777', 2, 40.0);
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO WORKS_ON(Essn, Pno, Hours) VALUES('987654321', 3, 20.0);
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO WORKS_ON(Essn, Pno, Hours) VALUES('666884444', 10, 10.0);
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO WORKS_ON(Essn, Pno, Hours) VALUES('453453453', 20, 30.0);
Query OK, 1 row affected (0.00 sec)

mysql>
mysql> INSERT INTO DEPENDENT(Essn, Dependent_name, Sex, Bdate, Relationship) VALUES('123456789', 'John Smith', 'M', '1965-01-09', '731
 Fondren, Houston, TX');
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO DEPENDENT(Essn, Dependent_name, Sex, Bdate, Relationship) VALUES('333445555', 'Franklin Wong', 'M', '1955-12-08', '
638 Voss, Houston, TX');
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO DEPENDENT(Essn, Dependent_name, Sex, Bdate, Relationship) VALUES('999887777', 'Alicia Zelaya', 'F', '1968-01-19', '
3321 Castle, Spring, TX');
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO DEPENDENT(Essn, Dependent_name, Sex, Bdate, Relationship) VALUES('987654321', 'Jennifer Wallace', 'F', '1941-06-20'
, '291 Berry, Bellaire, TX');
ERROR 1406 (22001): Data too long for column 'Dependent_name' at row 1
mysql> INSERT INTO DEPENDENT(Essn,Dependent_name, Sex, Bdate, Relationship) VALUES('666884444', 'Ramesh Narayan', 'M', '1962-09-15', '
975 Fire Oak, Humble, TX');
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO DEPENDENT(Essn, Dependent_name, Sex, Bdate, Relationship) VALUES('453453453', 'Joyce English', 'F', '1972-07-31', '
5631 Rice, Houston, TX');
Query OK, 1 row affected (0.00 sec)
```

```
mysql> -- Problem Statement Answers
mysql>
mysql> -- 1)    Find the sum of the salaries of all employees, the maximum salary, the minimum salary, and the average salary.
mysql> SELECT SUM(Salary) AS "Sum of Salaries", MAX(Salary) AS "Max Salary", MIN(Salary) AS "Min Salary", AVG(Salary) AS "Average Sala
ry" FROM EMPLOYEE;
+-----------------+------------+------------+----------------+
| Sum of Salaries | Max Salary | Min Salary | Average Salary |
+-----------------+------------+------------+----------------+
|       281000.00 |   55000.00 |   25000.00 |   35125.000000 |
+-----------------+------------+------------+----------------+
1 row in set (0.00 sec)

mysql>
mysql> -- 2)    Find the sum of the salaries of all employees of the 'Research' department, as well as the maximum salary, the minimum
 salary, and the average salary in this department.
mysql> SELECT SUM(Salary) AS "Sum of Salaries", MAX(Salary) AS "Max Salary", MIN(Salary) AS "Min Salary", AVG(Salary) AS "Average Sala
ry" FROM EMPLOYEE WHERE Dno = 5;
+-----------------+------------+------------+----------------+
| Sum of Salaries | Max Salary | Min Salary | Average Salary |
+-----------------+------------+------------+----------------+
|       106000.00 |   43000.00 |   25000.00 |   35333.333333 |
+-----------------+------------+------------+----------------+
1 row in set (0.00 sec)

mysql> -- 3)    Retrieve the total number of employees in the company.
mysql> SELECT COUNT(*) AS "Total Number of Employees" FROM EMPLOYEE;
+--------------------------+
| Total Number of Employees |
+--------------------------+
|                        8 |
+--------------------------+
1 row in set (0.01 sec)

mysql>
mysql> -- 4)    Retrieve the number of employees in the 'Research' department.
mysql> SELECT COUNT(*) AS "Number of Employees in Research" FROM EMPLOYEE
    -> WHERE Dno = 5;
+---------------------------------+
| Number of Employees in Research |
+---------------------------------+
|                               3 |
+---------------------------------+
1 row in set (0.00 sec)

mysql>
mysql> -- 5)    Count the number of distinct salary values in the database
mysql> SELECT COUNT(DISTINCT Salary) AS "Number of Distinct Salaries" FROM EMPLOYEE;
+----------------------------+
| Number of Distinct Salaries |
+----------------------------+
|                          6 |
+----------------------------+
1 row in set (0.00 sec)
```

```
mysql> -- 6)    For each department, retrieve the department number, the number of employees in the department, and their average sala
ry.
mysql> SELECT Dno, COUNT(*) AS "Number of Employees", AVG(Salary) AS "Average Salary" FROM EMPLOYEE
    -> GROUP BY Dno;
+------+---------------------+----------------+
| Dno  | Number of Employees | Average Salary |
+------+---------------------+----------------+
|    1 |                   3 |   35000.000000 |
|    5 |                   3 |   35333.333333 |
|    7 |                   2 |   35000.000000 |
+------+---------------------+----------------+
3 rows in set (0.00 sec)

mysql>
mysql> -- 7)    For each project, retrieve the project number, the project name, and the number of employees who work on that project.

mysql> SELECT Pnumber, Pname, COUNT(*) AS "Number of Employees" FROM PROJECT, WORKS_ON
    -> WHERE Pnumber = Pno GROUP BY Pnumber, Pname;
+---------+-----------------+---------------------+
| Pnumber | Pname           | Number of Employees |
+---------+-----------------+---------------------+
|      10 | Computerization |                   1 |
|       1 | ProductX        |                   2 |
|       2 | ProductY        |                   1 |
|       3 | ProductZ        |                   1 |
|      20 | Reorganization  |                   1 |
+---------+-----------------+---------------------+
5 rows in set (0.00 sec)

mysql> -- 8)    For each project on which more than two employees work, retrieve the project number, the project name, and the number
of employees who work on the project
mysql> SELECT Pnumber, Pname, COUNT(*) AS "Number of Employees" FROM PROJECT, WORKS_ON
    -> WHERE Pnumber = Pno
    -> GROUP BY Pnumber, Pname
    -> HAVING COUNT(*) > 2;
Empty set (0.00 sec)

mysql>
mysql> -- 9)    For each project, retrieve the project number, the project name, and the number of employees from department 5 who wor
k on the project.
mysql> SELECT Pnumber, Pname, COUNT(*) AS "Number of Employees" FROM PROJECT, WORKS_ON, EMPLOYEE
    -> WHERE Pnumber = Pno AND Essn = Ssn AND Dno = 5
    -> GROUP BY Pnumber, Pname;
+---------+-----------------+---------------------+
| Pnumber | Pname           | Number of Employees |
+---------+-----------------+---------------------+
|      10 | Computerization |                   1 |
|       3 | ProductZ        |                   1 |
|       2 | ProductY        |                   1 |
+---------+-----------------+---------------------+
3 rows in set (0.00 sec)

mysql>
mysql> -- 10)   For each department that has more than five employees, retrieve the department number and the number of its employees
who are making more than $40,000.
mysql> SELECT Dno, COUNT(*) AS "Number of Employees" FROM EMPLOYEE
    -> WHERE Salary > 40000
    -> GROUP BY Dno
    -> HAVING COUNT(*) > 5;
```

Preparatory Questions

Q1) Which of the following statements are TRUE about an SQL query? P: An SQL query can contain a HAVING clause even if it does not have a GROUP BY clause Q : An SQL query can contain a HAVING clause only if it has a GROUP BY clause R : All attributes used in the GROUP BY clause must appear in the SELECT clause S : Not all attributes used in the GROUP BY clause need to appear in the SELECT clause

1) P and R
2) P and S
3) Q and R
4) Q and S

Q2) Consider a database table T containing two columns X and Y each of type integer. After the creation of the table, one record (X=1, Y=1) is inserted in the table. Let MX and My denote the respective maximum values of X and Y among all records in the table at any point in time. Using MX and MY, new records are inserted in the table 128 times with X and Y values being MX+1, 2*MY+1 respectively. It may be noted that each time after the insertion, values of MX and MY change. What will be the output of the following SQL query after the steps mentioned above are carried out?

SELECT Y FROM T WHERE X=7;

1) 127
2) 255
3) 129
4) 257

Q3) The employee information in a company is stored in the relation

Employee (name, sex, salary, deptName)

Consider the following SQL query
select deptName

from Employee

where sex = 'M'

group by deptName

having avg (salary) > (select avg (salary) from Employee)

It returns the names of the department in which

1) the average salary is more than the average salary in the company
2) the average salary of male employees is more than the average salary of all male employees in the company
3) the average salary of male employees is more than the average salary of employees in the same department
4) the average salary of male employees is more than the average salary in the company

Q4) Which of the following is aggregate function in SQL?

1) Avg
2) Select
3) Ordered by
4) distinct

Q5) Observe the given SQL query and choose the correct option.

```
SELECT branch_name, COUNT (DISTINCT customer_name)
FROM depositor, account
WHERE depositor.account_number = account.account_number
GROUP BY branch_id
```

1) The query is syntactically correct but gives the wrong answer
2) The query is syntactically wrong
3) The query is syntactically correct and gives the correct answer
4) The query contains one or more wrongly named clauses.

Experiment No: 7

Student Name and Roll Number: Piyush Gambhir – 21CSU349
Semester /Section: Semester-IV – AIML-B (AL-3)
Link to Code: Piyush-Gambhir/NCU-CSL214-DBMS-Lab_Manual (github.com)
Date: 03.04.2023
Faculty Signature:
Marks:

Objective

To familiarize with nested SQL queries on the COMPANY database.

Program Outcome

- The students will understand how to devise independent and correlated nested queries.

Problem Statement

Consider the COMPANY database in Experiment 1 and execute the following queries:

1) Make a list of all project numbers for projects that involve an employee whose last name is 'Smith', either as a worker or as a manager of the department that controls the project.
2) Select the Essns of all employees who work on the same project and hours as some project that employee 'John Smith' (whose Ssn = '123456789') works on.
3) Return the names of employees whose salary is greater than the salary of all the employees in department 5
4) Retrieve the name of each employee who has a dependent with the same first name and is the same sex as the employee
5) Retrieve the names of employees who have no dependents
6) List the names of managers who have at least one dependent using EXISTS and NOT EXISTS functions
7) Retrieve the name of each employee who works on all the projects controlled by department number 5 using EXISTS and NOT EXISTS functions
8) Retrieve the names of all employees who have two or more dependents

9) Retrieves the names of all employees who work on only one project

Background Study

1) SQL allows queries that check whether an attribute value is NULL: IS or IS NOT NULL

2) Nested Queries: Complete select-from-where blocks within WHERE clause of another query
   - Nested Queries generally return a table (relation)

3) Comparison Operator IN:
   - Compares value *v* with a set (or multiset) of values *V*
   - Evaluates to TRUE if *v* is one of the elements in *V*

4) = ANY (or = SOME) Operator: Returns TRUE if the value *v* is equal to some value in the set *V* (equivalent to IN)

5) ALL Operator: (*v* > ALL *V*) returns TRUE if the value *v* is greater than *all* the values in the set (or multiset) *V*.
   - Other operators that can be combined with ANY (or SOME) and ALL: >, >=, <, <=, and <>

6) Possible ambiguity among attribute names if attributes of the same name exist—one in a relation in the FROM clause of the outer query, and another in a relation in the FROM clause of the nested query.
   - Thumb Rule: reference to an unqualified attribute refers to the relation declared in the innermost nested query.

7) Whenever a condition in the WHERE clause of a nested query references some attribute of a relation declared in the outer query, the two queries are said to be correlated.
   - In a correlated query, the nested query is evaluated once for each tuple (or combination of tuples) in the outer query

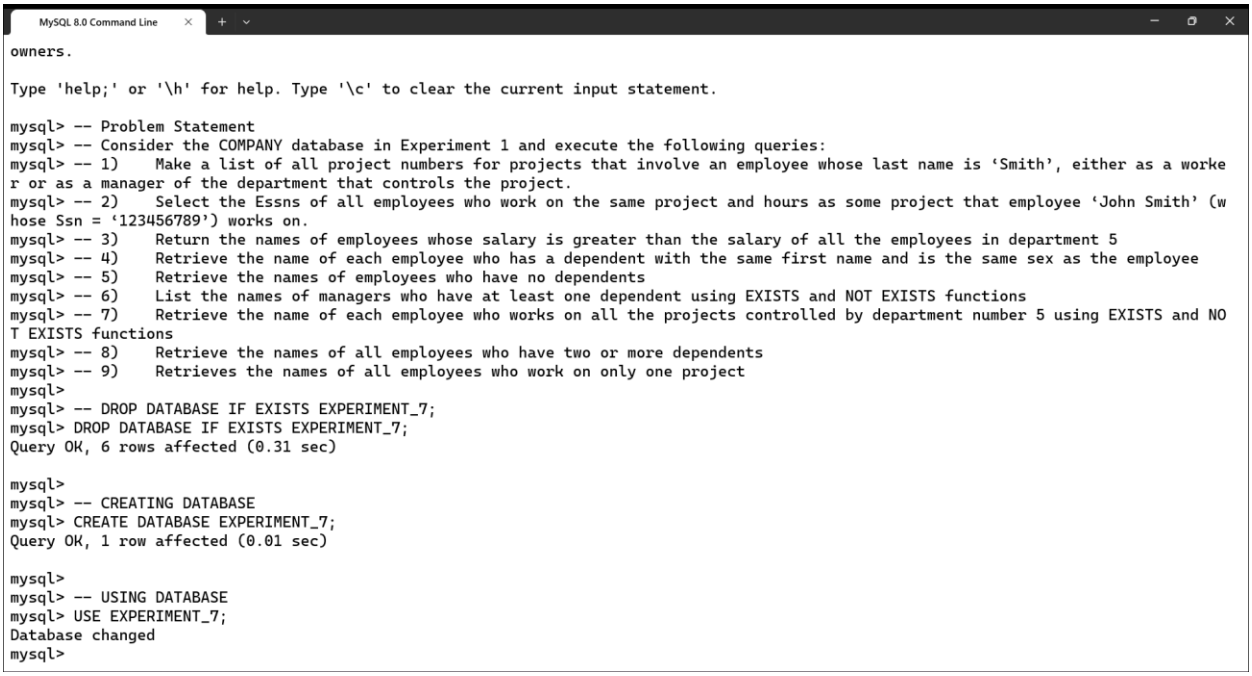
8) EXISTS function: Check whether the result of a correlated nested query is empty or not.

9) EXISTS and NOT EXISTS are typically used in conjunction with a correlated nested query.

10) EXISTS(Q): returns TRUE if there is at least one tuple in the result of the nested query Q, and it returns FALSE otherwise.

11) NOT EXISTS(Q): returns TRUE if there are no tuples in the result of nested query Q, and it returns FALSE otherwise.

Output: Screenshots




```
mysql> -- CREATING TABLES
mysql> CREATE TABLE Pack_grades(
    ->     Grade_id INT NOT NULL,
    ->     Grade_name VARCHAR(20) NOT NULL,
    ->     Min_price INT NOT NULL,
    ->     Max_price INT NOT NULL,
    ->     PRIMARY KEY(Grade_id)
    -> );
Query OK, 0 rows affected (0.01 sec)

mysql>
mysql> CREATE TABLE Customers(
    ->     Customer_id INT NOT NULL,
    ->     First_name VARCHAR(20) NOT NULL,
    ->     Last_name VARCHAR(20) NOT NULL,
    ->     Birth_date DATE NOT NULL,
    ->     Join_date DATE NOT NULL,
    ->     City VARCHAR(20) NOT NULL,
    ->     Pack_id INT,
    ->     State VARCHAR(20) NOT NULL,
    ->     PRIMARY KEY(Customer_id)
    -> );
Query OK, 0 rows affected (0.01 sec)
```

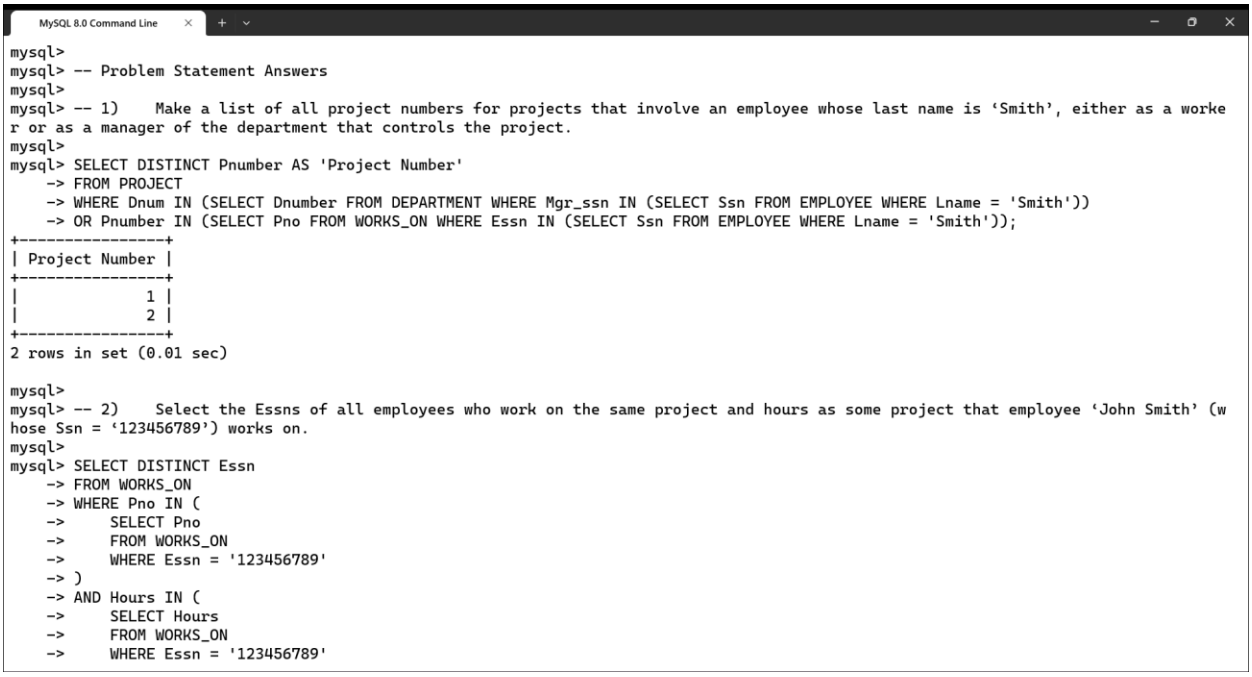
```
mysql> CREATE TABLE Packages(
    ->     Pack_id INT NOT NULL,
    ->     Speed INT NOT NULL,
    ->     Start_date DATE NOT NULL,
    ->     Monthly_payment INT NOT NULL,
    ->     Sector_id INT,
    ->     PRIMARY KEY(Pack_id)
    -> );
Query OK, 0 rows affected (0.01 sec)

mysql>
mysql> CREATE TABLE Sectors(
    ->     Sector_id INT NOT NULL,
    ->     Sector_name VARCHAR(20) NOT NULL,
    ->     PRIMARY KEY(Sector_id)
    -> );
Query OK, 0 rows affected (0.01 sec)

mysql>
mysql> ALTER TABLE Customers ADD
    -> FOREIGN KEY(Pack_id) REFERENCES Packages(Pack_id);
Query OK, 0 rows affected (0.05 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql>
mysql> ALTER TABLE Packages ADD
    -> FOREIGN KEY(Sector_id) REFERENCES Sectors(Sector_id);
Query OK, 0 rows affected (0.04 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

```
mysql> -- INSERTING DATA
mysql>
mysql> INSERT INTO DEPARTMENT(Dname, Dnumber, Mgr_start_date, Mgr_ssn) VALUES('Research', 5, '1988-05-22', NULL);
Query OK, 1 row affected (0.01 sec)

mysql> INSERT INTO DEPARTMENT(Dname, Dnumber, Mgr_start_date, Mgr_ssn) VALUES('Administration', 4, '1995-01-01', NULL);
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO DEPARTMENT(Dname, Dnumber, Mgr_start_date, Mgr_ssn) VALUES('Headquarters', 1, '1981-06-19', NULL);
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO DEPARTMENT(Dname, Dnumber, Mgr_start_date, Mgr_ssn) VALUES('Software', 7, '1998-06-19', NULL);
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO DEPARTMENT(Dname, Dnumber, Mgr_start_date, Mgr_ssn) VALUES('Services', 3, '1985-01-01', NULL);
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO DEPARTMENT(Dname, Dnumber, Mgr_start_date, Mgr_ssn) VALUES('Finance', 8, '1995-01-01', NULL);
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO DEPARTMENT(Dname, Dnumber, Mgr_start_date, Mgr_ssn) VALUES('Public Relations', 2, '1993-07-08', NULL);
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO DEPARTMENT(Dname, Dnumber, Mgr_start_date, Mgr_ssn) VALUES('Production', 6, '1996-05-01', NULL);
Query OK, 1 row affected (0.00 sec)

mysql>
mysql> INSERT INTO EMPLOYEE(Fname, Minit, Lname, Ssn, Bdate, Address, Sex, Salary, Super_ssn, Dno) VALUES('John', 'B', 'Smith', '12345
6789', '1965-01-09', '731 Fondren, Houston, TX', 'M', 30000.00, NULL, NULL);
Query OK, 1 row affected (0.01 sec)

mysql> INSERT INTO EMPLOYEE(Fname, Minit, Lname, Ssn, Bdate, Address, Sex, Salary, Super_ssn, Dno) VALUES('Franklin', 'T', 'Wong', '33
3445555', '1955-12-08', '638 Voss, Houston, TX', 'M', 40000.00, NULL, NULL);
```

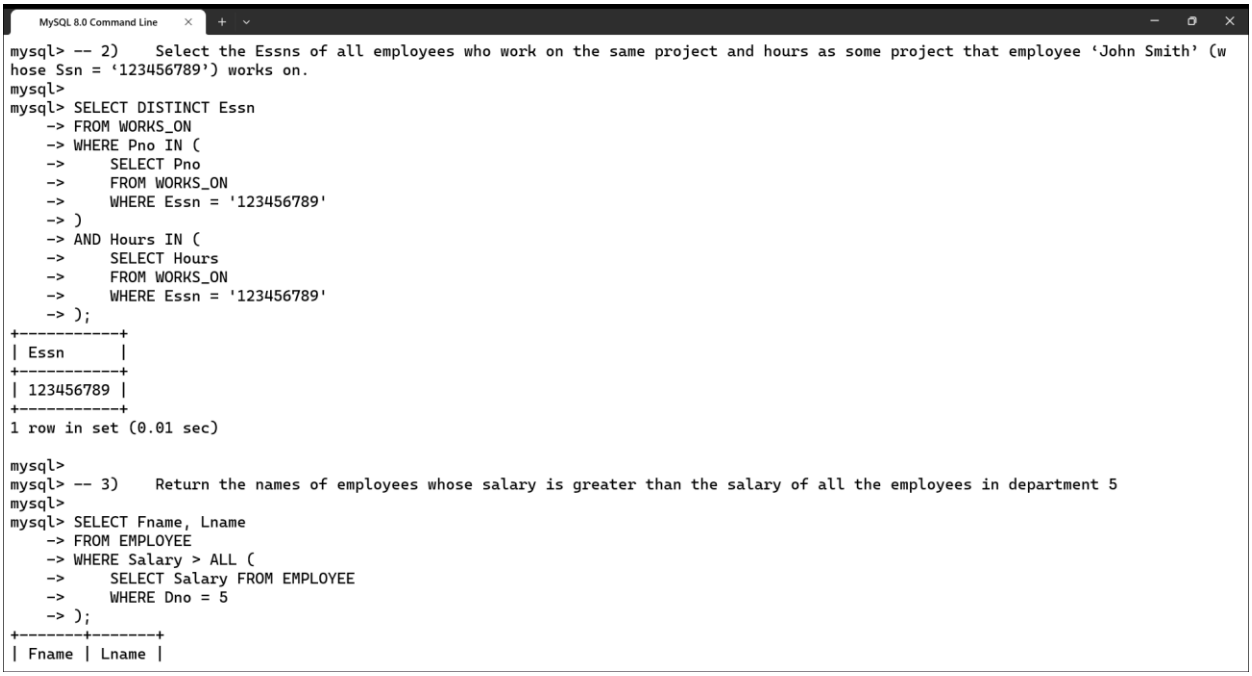
```
mysql> INSERT INTO EMPLOYEE(Fname, Minit, Lname, Ssn, Bdate, Address, Sex, Salary, Super_ssn, Dno) VALUES('Alicia', 'J', 'Zelaya', '99
98887777', '1968-01-19', '3321 Castle, Spring, TX', 'F', 25000.00, NULL, NULL);
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO EMPLOYEE(Fname, Minit, Lname, Ssn, Bdate, Address, Sex, Salary, Super_ssn, Dno) VALUES('Jennifer', 'S', 'Wallace',
'987654321', '1941-06-20', '291 Berry, Bellaire, TX', 'F', 43000.00, NULL, NULL);
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO EMPLOYEE(Fname, Minit, Lname, Ssn, Bdate, Address, Sex, Salary, Super_ssn, Dno) VALUES('Ramesh', 'K', 'Narayan', '6
66884444', '1962-09-15', '975 Fire Oak, Humble, TX', 'M', 38000.00, NULL, NULL);
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO EMPLOYEE(Fname, Minit, Lname, Ssn, Bdate, Address, Sex, Salary, Super_ssn, Dno) VALUES('Joyce', 'A', 'English', '45
3453453', '1972-07-31', '5631 Rice, Houston, TX', 'F', 25000.00, NULL, NULL);
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO EMPLOYEE(Fname, Minit, Lname, Ssn, Bdate, Address, Sex, Salary, Super_ssn, Dno) VALUES( 'Ahmad', 'V', 'Jabbar', '88
8665555', '1969-03-29', '980 Dallas, Houston, TX', 'M', 25000.00, NULL, NULL);
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO EMPLOYEE(Fname, Minit, Lname, Ssn, Bdate, Address, Sex, Salary, Super_ssn, Dno) VALUES('James', 'E', 'Borg', '88866
4555', '1937-11-10', '450 Stone, Houston, TX', 'M', 55000.00, NULL, NULL);
Query OK, 1 row affected (0.00 sec)

mysql>
mysql> UPDATE DEPARTMENT SET Mgr_ssn = '123456789' WHERE Dnumber = 5;
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> UPDATE DEPARTMENT SET Mgr_ssn = '333445555' WHERE Dnumber = 4;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0




mysql> UPDATE DEPARTMENT SET Mgr_ssn = '999887777' WHERE Dnumber = 1;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> UPDATE DEPARTMENT SET Mgr_ssn = '987654321' WHERE Dnumber = 7;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> UPDATE DEPARTMENT SET Mgr_ssn = '666884444' WHERE Dnumber = 3;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> UPDATE DEPARTMENT SET Mgr_ssn = '453453453' WHERE Dnumber = 8;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> UPDATE DEPARTMENT SET Mgr_ssn = '888664555' WHERE Dnumber = 2;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> UPDATE DEPARTMENT SET Mgr_ssn = '888665555' WHERE Dnumber = 6;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql>
mysql> UPDATE EMPLOYEE SET Super_ssn = '987654321', Dno = 7 WHERE Ssn = '123456789';
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> UPDATE EMPLOYEE SET Super_ssn = '987654321', Dno = 7 WHERE Ssn = '333445555';
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

```
mysql> UPDATE EMPLOYEE SET Super_ssn = '333445555', Dno = 5 WHERE Ssn = '999887777';
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> UPDATE EMPLOYEE SET Super_ssn = '333445555', Dno = 5 WHERE Ssn = '987654321';
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> UPDATE EMPLOYEE SET Super_ssn = '333445555', Dno = 5 WHERE Ssn = '666884444';
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> UPDATE EMPLOYEE SET Super_ssn = '453453453', Dno = 1 WHERE Ssn = '453453453';
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> UPDATE EMPLOYEE SET Super_ssn = '453453453', Dno = 1 WHERE Ssn = '888665555';
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> UPDATE EMPLOYEE SET Super_ssn = '453453453', Dno = 1 WHERE Ssn = '888664555';
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql>
mysql> INSERT INTO DEPARTMENT_LOCATIONS(DNumber, DLocation) VALUES(1, 'Stafford');
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO DEPARTMENT_LOCATIONS(DNumber, DLocation) VALUES(1, 'Bellaire');
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO DEPARTMENT_LOCATIONS(DNumber, DLocation) VALUES(2, 'Houston');
Query OK, 1 row affected (0.00 sec)




mysql> INSERT INTO DEPARTMENT_LOCATIONS(DNumber, DLocation) VALUES(2, 'Sugar Land');
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO DEPARTMENT_LOCATIONS(DNumber, DLocation) VALUES(3, 'Bellaire');
Query OK, 1 row affected (0.00 sec)

mysql>
mysql> INSERT INTO PROJECT(Pname, Pnumber, Plocation, Dnum) VALUES('ProductX', 1, 'Bellaire', 5);
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO PROJECT(Pname, Pnumber, Plocation, Dnum) VALUES('ProductY', 2, 'Stafford', 5);
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO PROJECT(Pname, Pnumber, Plocation, Dnum) VALUES('ProductZ', 3, 'Houston', 6);
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO PROJECT(Pname, Pnumber, Plocation, Dnum) VALUES('Computerization', 10, 'Stafford', 1);
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO PROJECT(Pname, Pnumber, Plocation, Dnum) VALUES('Reorganization', 20, 'Houston', 4);
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO PROJECT(Pname, Pnumber, Plocation, Dnum) VALUES('Newbenefits', 30, 'Stafford', 4);
Query OK, 1 row affected (0.00 sec)

mysql>
mysql> INSERT INTO WORKS_ON(Essn, Pno, Hours) VALUES('123456789', 1, 32.5);
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO WORKS_ON(Essn, Pno, Hours) VALUES('333445555', 1, 7.5);
Query OK, 1 row affected (0.00 sec)
```

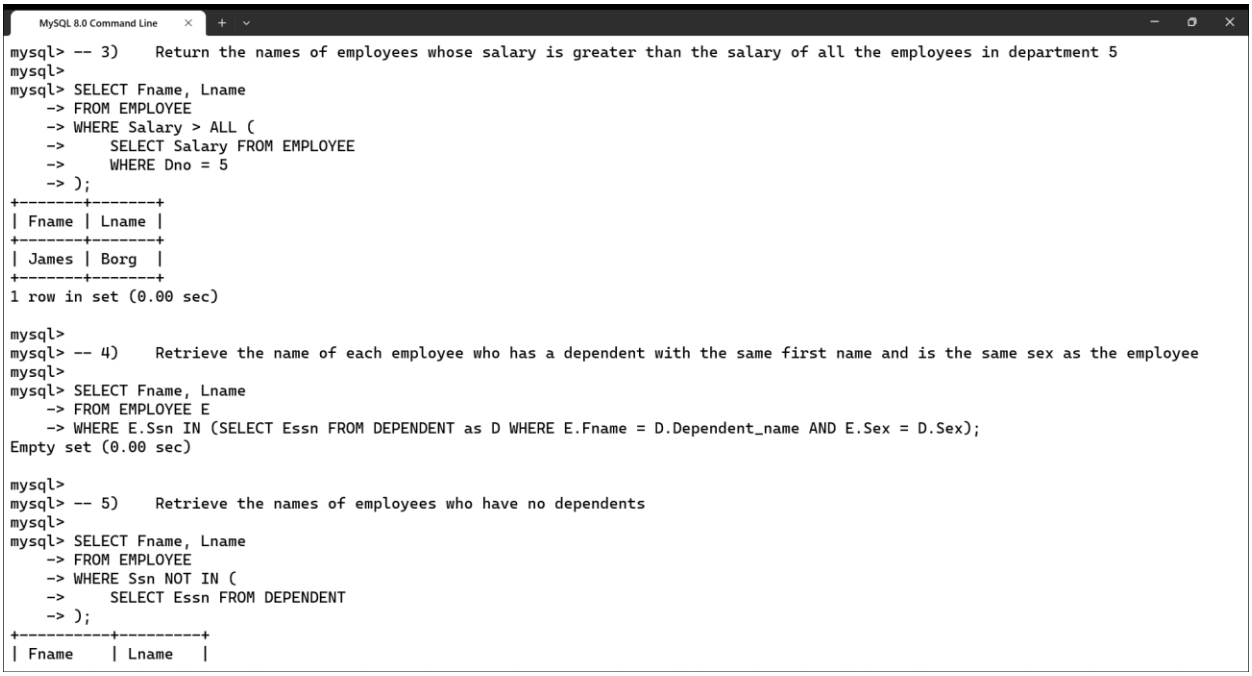
```
mysql> INSERT INTO WORKS_ON(Essn, Pno, Hours) VALUES('999887777', 2, 40.0);
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO WORKS_ON(Essn, Pno, Hours) VALUES('987654321', 3, 20.0);
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO WORKS_ON(Essn, Pno, Hours) VALUES('666884444', 10, 10.0);
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO WORKS_ON(Essn, Pno, Hours) VALUES('453453453', 20, 30.0);
Query OK, 1 row affected (0.00 sec)

mysql>
mysql> INSERT INTO DEPENDENT(Essn, Dependent_name, Sex, Bdate, Relationship) VALUES('123456789', 'John Smith', 'M', '1965-01-09', '731
 Fondren, Houston, TX');
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO DEPENDENT(Essn, Dependent_name, Sex, Bdate, Relationship) VALUES('333445555', 'Franklin Wong', 'M', '1955-12-08', '
638 Voss, Houston, TX');
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO DEPENDENT(Essn, Dependent_name, Sex, Bdate, Relationship) VALUES('999887777', 'Alicia Zelaya', 'F', '1968-01-19', '
3321 Castle, Spring, TX');
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO DEPENDENT(Essn, Dependent_name, Sex, Bdate, Relationship) VALUES('987654321', 'Jennifer Wallace', 'F', '1941-06-20'
, '291 Berry, Bellaire, TX');
ERROR 1406 (22001): Data too long for column 'Dependent_name' at row 1
mysql> INSERT INTO DEPENDENT(Essn,Dependent_name, Sex, Bdate, Relationship) VALUES('666884444', 'Ramesh Narayan', 'M', '1962-09-15', '
975 Fire Oak, Humble, TX');
Query OK, 1 row affected (0.00 sec)
```



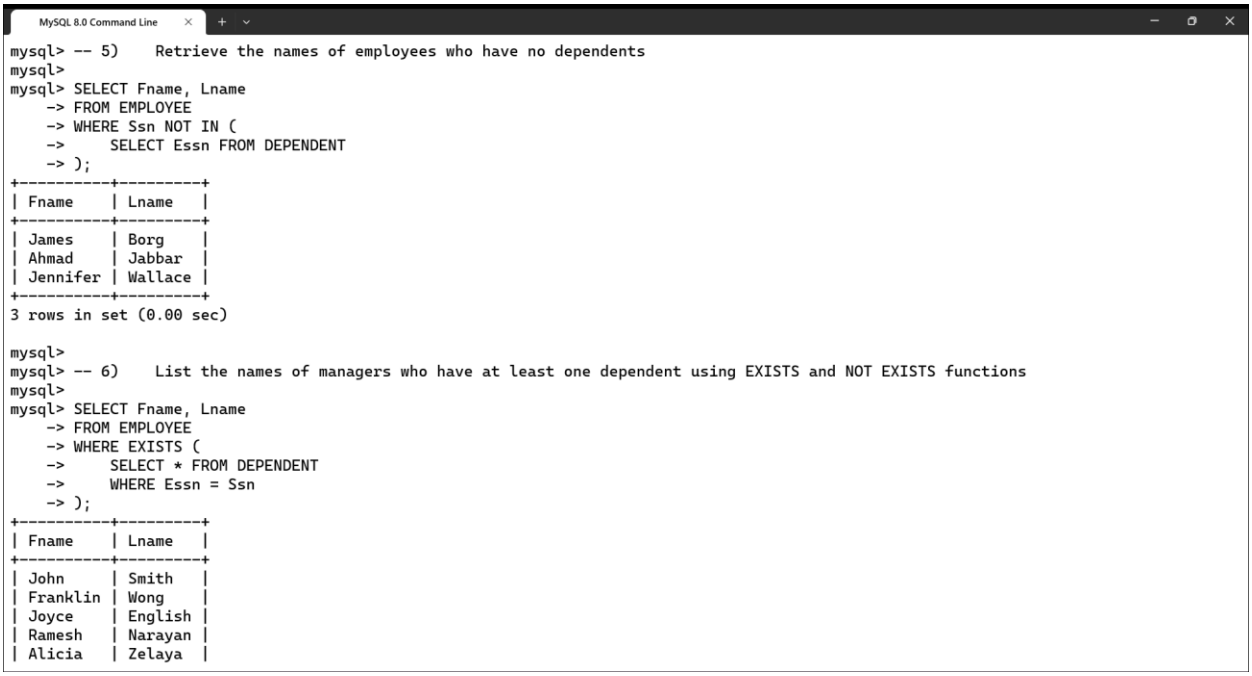
```
MySQL 8.0 Command Line

mysql>
mysql> -- Problem Statement Answers
mysql>
mysql> -- 1)    Make a list of all project numbers for projects that involve an employee whose last name is ‘Smith’, either as a worke
r or as a manager of the department that controls the project.
mysql>
mysql> SELECT DISTINCT Pnumber AS 'Project Number'
    -> FROM PROJECT
    -> WHERE Dnum IN (SELECT Dnumber FROM DEPARTMENT WHERE Mgr_ssn IN (SELECT Ssn FROM EMPLOYEE WHERE Lname = 'Smith'))
    -> OR Pnumber IN (SELECT Pno FROM WORKS_ON WHERE Essn IN (SELECT Ssn FROM EMPLOYEE WHERE Lname = 'Smith'));
+----------------+
| Project Number |
+----------------+
|              1 |
|              2 |
+----------------+
2 rows in set (0.01 sec)

mysql>
mysql> -- 2)    Select the Essns of all employees who work on the same project and hours as some project that employee ‘John Smith’ (w
hose Ssn = ‘123456789’) works on.
mysql>
mysql> SELECT DISTINCT Essn
    -> FROM WORKS_ON
    -> WHERE Pno IN (
    ->     SELECT Pno
    ->     FROM WORKS_ON
    ->     WHERE Essn = '123456789'
    -> )
    -> AND Hours IN (
    ->     SELECT Hours
    ->     FROM WORKS_ON
    ->     WHERE Essn = '123456789'
```



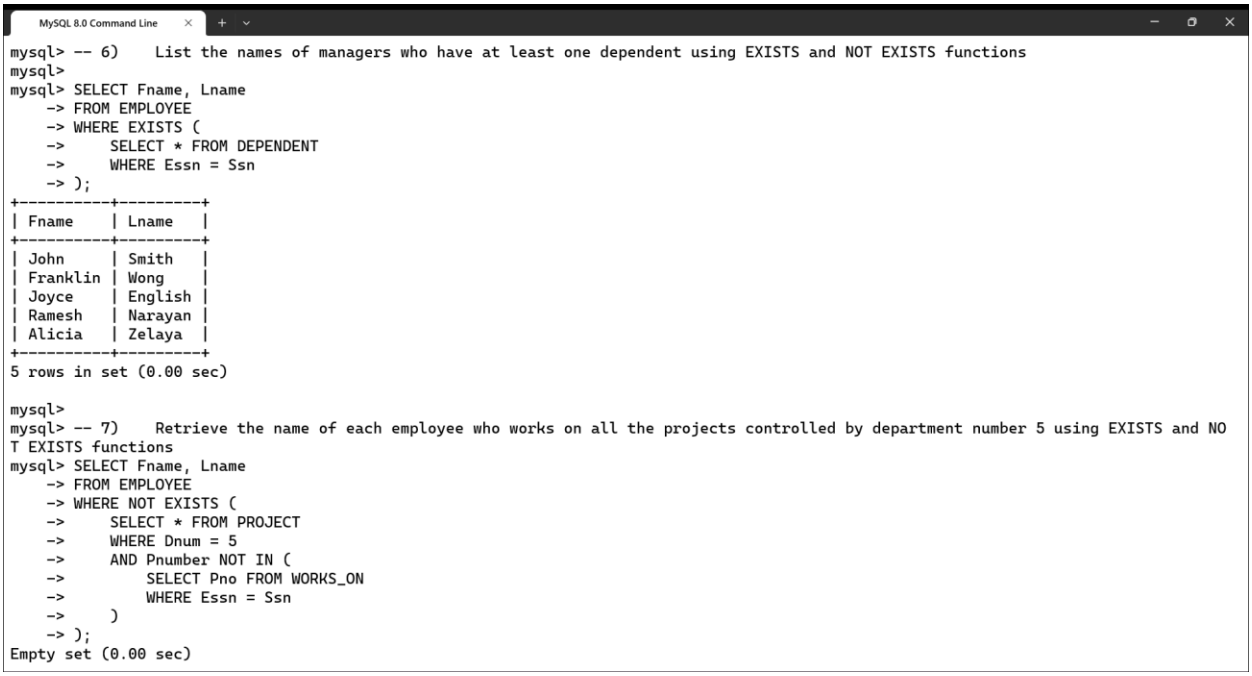
```
MySQL 8.0 Command Line
mysql> -- 2)    Select the Essns of all employees who work on the same project and hours as some project that employee 'John Smith' (w
hose Ssn = '123456789') works on.
mysql>
mysql> SELECT DISTINCT Essn
    -> FROM WORKS_ON
    -> WHERE Pno IN (
    ->     SELECT Pno
    ->     FROM WORKS_ON
    ->     WHERE Essn = '123456789'
    -> )
    -> AND Hours IN (
    ->     SELECT Hours
    ->     FROM WORKS_ON
    ->     WHERE Essn = '123456789'
    -> );
+-----------+
| Essn      |
+-----------+
| 123456789 |
+-----------+
1 row in set (0.01 sec)

mysql>
mysql> -- 3)    Return the names of employees whose salary is greater than the salary of all the employees in department 5
mysql>
mysql> SELECT Fname, Lname
    -> FROM EMPLOYEE
    -> WHERE Salary > ALL (
    ->     SELECT Salary FROM EMPLOYEE
    ->     WHERE Dno = 5
    -> );
+-------+-------+
| Fname | Lname |
```



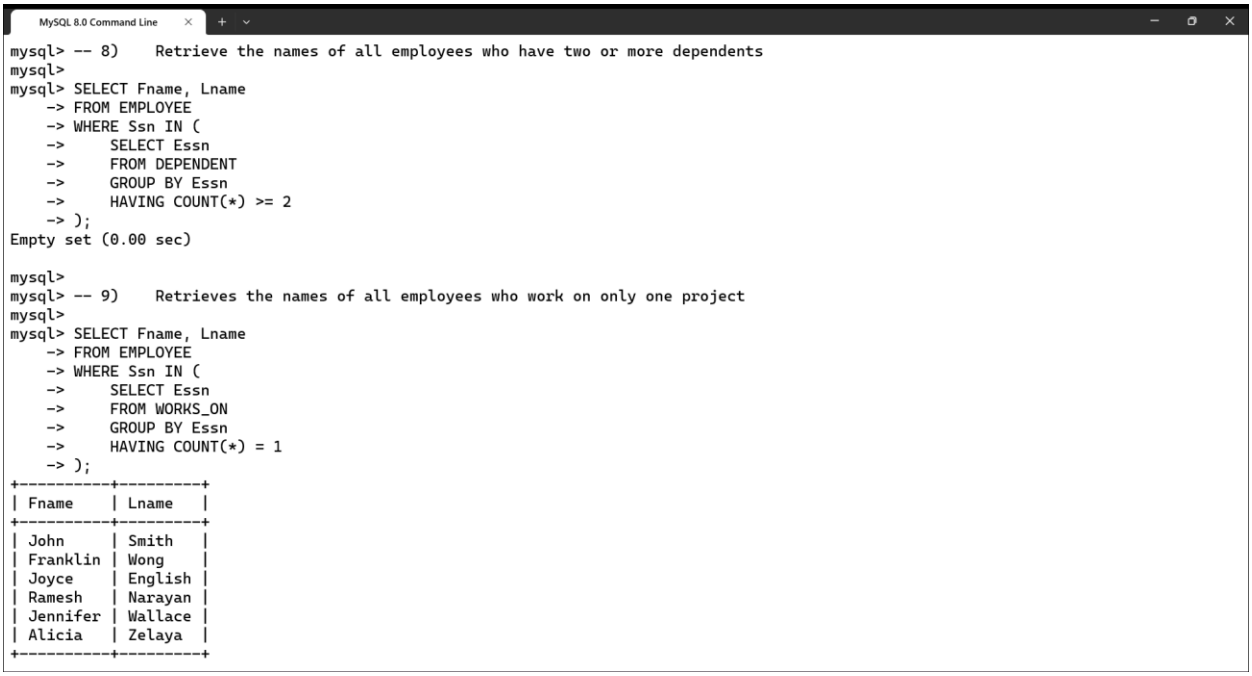
```
MySQL 8.0 Command Line
mysql> -- 3)    Return the names of employees whose salary is greater than the salary of all the employees in department 5
mysql>
mysql> SELECT Fname, Lname
    -> FROM EMPLOYEE
    -> WHERE Salary > ALL (
    ->     SELECT Salary FROM EMPLOYEE
    ->     WHERE Dno = 5
    -> );
+-------+-------+
| Fname | Lname |
+-------+-------+
| James | Borg  |
+-------+-------+
1 row in set (0.00 sec)

mysql>
mysql> -- 4)    Retrieve the name of each employee who has a dependent with the same first name and is the same sex as the employee
mysql>
mysql> SELECT Fname, Lname
    -> FROM EMPLOYEE E
    -> WHERE E.Ssn IN (SELECT Essn FROM DEPENDENT as D WHERE E.Fname = D.Dependent_name AND E.Sex = D.Sex);
Empty set (0.00 sec)

mysql>
mysql> -- 5)    Retrieve the names of employees who have no dependents
mysql>
mysql> SELECT Fname, Lname
    -> FROM EMPLOYEE
    -> WHERE Ssn NOT IN (
    ->     SELECT Essn FROM DEPENDENT
    -> );
+----------+---------+
| Fname    | Lname   |
```



```
MySQL 8.0 Command Line
mysql> -- 5)    Retrieve the names of employees who have no dependents
mysql>
mysql> SELECT Fname, Lname
    -> FROM EMPLOYEE
    -> WHERE Ssn NOT IN (
    ->     SELECT Essn FROM DEPENDENT
    -> );
+----------+---------+
| Fname    | Lname   |
+----------+---------+
| James    | Borg    |
| Ahmad    | Jabbar  |
| Jennifer | Wallace |
+----------+---------+
3 rows in set (0.00 sec)

mysql>
mysql> -- 6)    List the names of managers who have at least one dependent using EXISTS and NOT EXISTS functions
mysql>
mysql> SELECT Fname, Lname
    -> FROM EMPLOYEE
    -> WHERE EXISTS (
    ->     SELECT * FROM DEPENDENT
    ->     WHERE Essn = Ssn
    -> );
+----------+---------+
| Fname    | Lname   |
+----------+---------+
| John     | Smith   |
| Franklin | Wong    |
| Joyce    | English |
| Ramesh   | Narayan |
| Alicia   | Zelaya  |
```



```
MySQL 8.0 Command Line
mysql> -- 6)    List the names of managers who have at least one dependent using EXISTS and NOT EXISTS functions
mysql>
mysql> SELECT Fname, Lname
    -> FROM EMPLOYEE
    -> WHERE EXISTS (
    ->     SELECT * FROM DEPENDENT
    ->     WHERE Essn = Ssn
    -> );
+----------+---------+
| Fname    | Lname   |
+----------+---------+
| John     | Smith   |
| Franklin | Wong    |
| Joyce    | English |
| Ramesh   | Narayan |
| Alicia   | Zelaya  |
+----------+---------+
5 rows in set (0.00 sec)

mysql>
mysql> -- 7)    Retrieve the name of each employee who works on all the projects controlled by department number 5 using EXISTS and NO
T EXISTS functions
mysql> SELECT Fname, Lname
    -> FROM EMPLOYEE
    -> WHERE NOT EXISTS (
    ->     SELECT * FROM PROJECT
    ->     WHERE Dnum = 5
    ->     AND Pnumber NOT IN (
    ->         SELECT Pno FROM WORKS_ON
    ->         WHERE Essn = Ssn
    ->     )
    -> );
Empty set (0.00 sec)
```


MySQL 8.0 Command Line
mysql> -- 8)   Retrieve the names of all employees who have two or more dependents
mysql>
mysql> SELECT Fname, Lname
    -> FROM EMPLOYEE
    -> WHERE Ssn IN (
    ->   SELECT Essn
    ->   FROM DEPENDENT
    ->   GROUP BY Essn
    ->   HAVING COUNT(*) >= 2
    ->);
Empty set (0.00 sec)
mysql>
mysql> -- 9)   Retrieves the names of all employees who work on only one project
mysql>
mysql> SELECT Fname, Lname
    -> FROM EMPLOYEE
    -> WHERE Ssn IN (
    ->   SELECT Essn
    ->   FROM WORKS_ON
    ->   GROUP BY Essn
    ->   HAVING COUNT(*) = 1
    ->);
+----------+---------+
| Fname | Lname |
+----------+---------+
John	Smith
Franklin	Wong
Joyce	English
Ramesh	Narayan
Jennifer	Wallace
Alicia	Zelaya
+----------+---------+


MySQL 8.0 Command Line
    -> WHERE Ssn IN (
    ->   SELECT Essn
    ->   FROM DEPENDENT
    ->   GROUP BY Essn
    ->   HAVING COUNT(*) >= 2
    ->);
Empty set (0.00 sec)
mysql>
mysql> -- 9)   Retrieves the names of all employees who work on only one project
mysql>
mysql> SELECT Fname, Lname
    -> FROM EMPLOYEE
    -> WHERE Ssn IN (
    ->   SELECT Essn
    ->   FROM WORKS_ON
    ->   GROUP BY Essn
    ->   HAVING COUNT(*) = 1
    ->);
+----------+---------+
| Fname | Lname |
+----------+---------+
John	Smith
Franklin	Wong
Joyce	English
Ramesh	Narayan
Jennifer	Wallace
Alicia	Zelaya
+----------+---------+
6 rows in set (0.00 sec)
mysql>

Preparatory Questions

Q1) Database table by name Loan_Records is given below.

Borrower	Bank_Manager	Loan_Amount
Ramesh	Sunderajan	10000.00
Suresh	Ramgopal	5000.00
Mahesh	Sunderajan	7000.00

What is the output of the following SQL query?

```
SELECT Count(*)
FROM  ( ( SELECT Borrower, Bank_Manager
        FROM Loan_Records) AS S
        NATURAL JOIN ( SELECT Bank_Manager, Loan_Amount
                FROM Loan_Records) AS T );
```

1) 3
2) 9
3) 5
4) 6

Q2) A relational schema for a train reservation database is given below. Passenger (pid, pname, age) Reservation (pid, class, tid)

Table: Passenger

pid	pname	age
0	Sachin	65
1	Rahul	66
2	Sourav	67
3	Anil	69

Table : Reservation

pid	class	tid
0	AC	8200
1	AC	8201

2 SC 8201
5 AC 8203
1 SC 8204
3 AC 8202

What pids are returned by the following SQL query for the above instance of the tables?

SELECT pid

FROM Reservation ,

WHERE class 'AC' AND

*EXISTS (SELECT **

FROM Passenger

WHERE age > 65 AND

Passenger. pid = Reservation.pid)

1) 1,0
2) 1,2
3) 1,3
4) 1,5

Q3) Consider the following relational schema:

Suppliers(sid:integer, sname:string, city:string, street:string)
Parts(pid:integer, pname:string, color:string)
Catalog(sid:integer, pid:integer, cost:real)

Consider the following relational query on the above database:

SELECT S.sname

FROM Suppliers S

WHERE S.sid NOT IN (SELECT C.sid

FROM Catalog C

WHERE C.pid NOT IN (SELECT P.pid

FROM Parts P

WHERE P.color<> 'blue'))

Assume that relations corresponding to the above schema are not empty. Which one of the following is the correct interpretation of the above query?

1) Find the names of all suppliers who have supplied a non-blue part.
2) Find the names of all suppliers who have not supplied a non-blue part.
3) Find the names of all suppliers who have supplied only blue parts.
4) Find the names of all suppliers who have not supplied only blue parts.
5) None

Q4) Consider the table employee(empId, name, department, salary) and the two queries Q1 ,Q2 below. Assuming that department 5 has more than one employee, and we want to find the employees who get higher salary than anyone in the department 5, which one of the statements is TRUE for any arbitrary employee table?

Query1: *Select e.empId*

From employee e

Where not exists

*(Select * From employee s where s.department = "5" and s.salary >=e.salary)*

Query2: *Select e.empId*

From employee e

Where e.salary > Any

(Select distinct salary From employee s Where s.department = "5")

1) Q1 is the correct query
2) Q2 is the correct query
3) Both Q1 and Q2 produce the same answer.
4) Neither Q1 nor Q2 is the correct query

Q5) Consider the following relational schema:

employee(empId, empName, empDept)
customer(custId, custName, salesRepId, rating)

salesRepId is a foreign key referring to empId of the employee relation. Assume that each employee makes a sale to at least one customer. What does the following query return?

```
SELECT empName
    FROM employee E
    WHERE NOT EXISTS ( SELECT custId
            FROM customer C
            WHERE C.salesRepId = E.empId
             AND C.rating <> `GOOD`);
```

1) Names of all the employees with at least one of their customers having a 'GOOD' rating.
2) Names of all the employees with at most one of their customers having a 'GOOD' rating.
3) Names of all the employees with none of their customers having a 'GOOD' rating.
4) Names of all the employees with all their customers having a 'GOOD' rating.

Experiment 8

Student Name and Roll Number: Piyush Gambhir – 21CSU349
Semester /Section: Semester-IV – AIML-B (AL-3)
Link to Code: Piyush-Gambhir/NCU-CSL214-DBMS-Lab_Manual (github.com)
Date: 17.04.2023
Faculty Signature:
Marks:

Objective

Identifying contrast between Relational Databases and NoSQL, thereby recognizing their applications.

Program Outcome

- The students will install MongoDB shell and familiarize themselves with it.

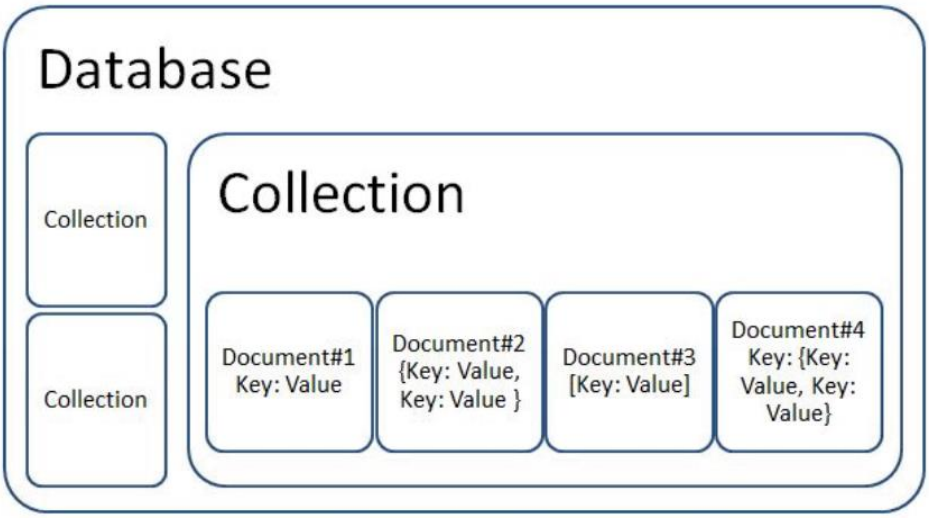
Problem Statement

MongoDB installation and shell familiarity.

Background Study

MongoDB is a Schema less database. A database in MongoDB contains collections and inside collections we have documents.

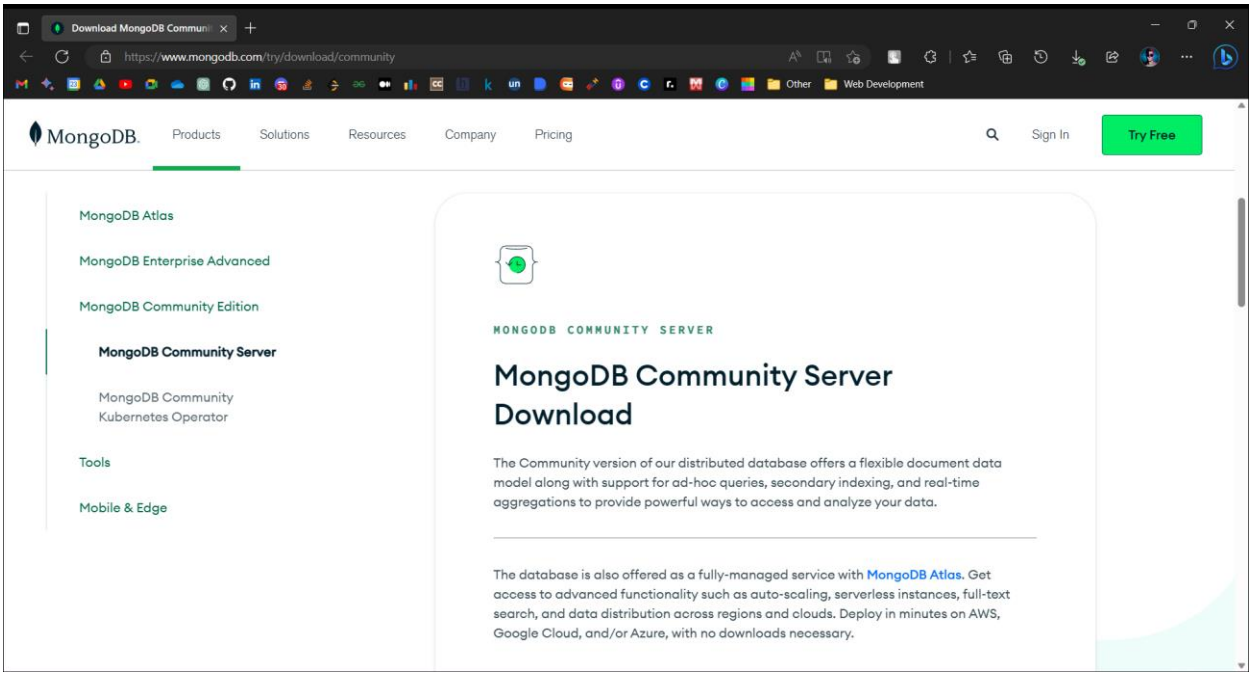
- Database is a physical container for collections. A single MongoDB server typically has multiple databases.
- Collection is a group of MongoDB documents.
- Documents within a collection can have different fields. A document is a set of key-value pairs and have dynamic schema. Dynamic schema means that documents in the same collection do not need to have the same set of fields or structure, and common fields in a collection's documents may hold different types of data.

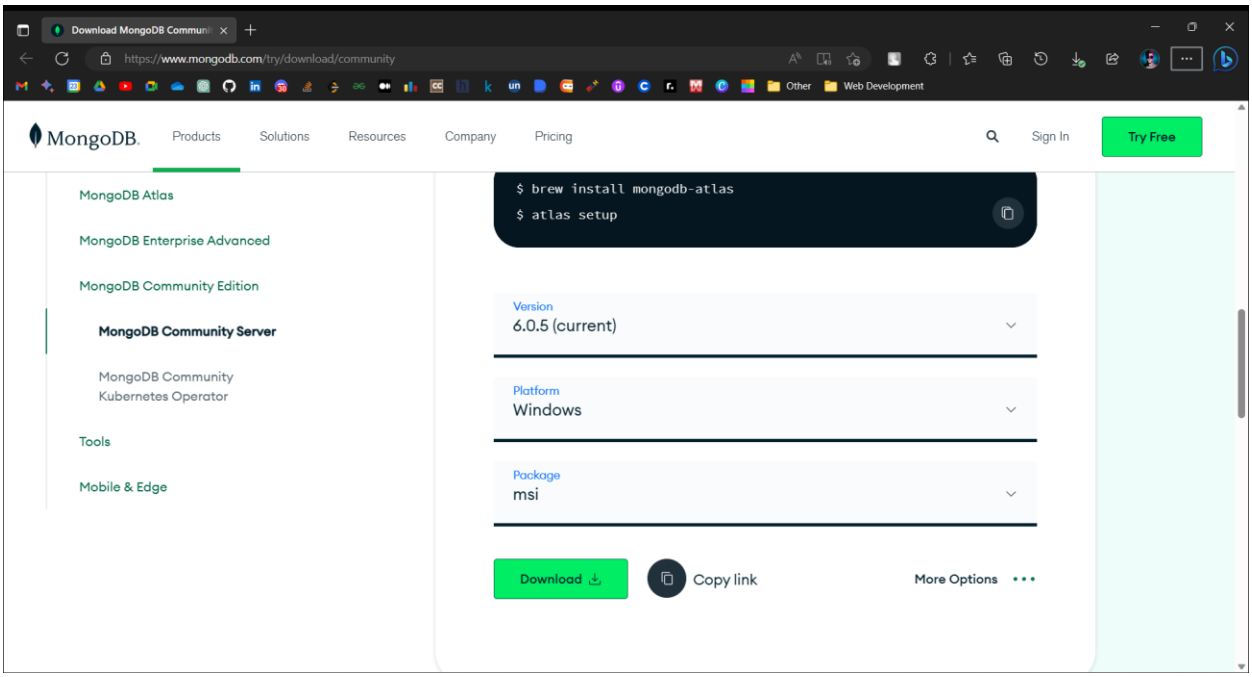


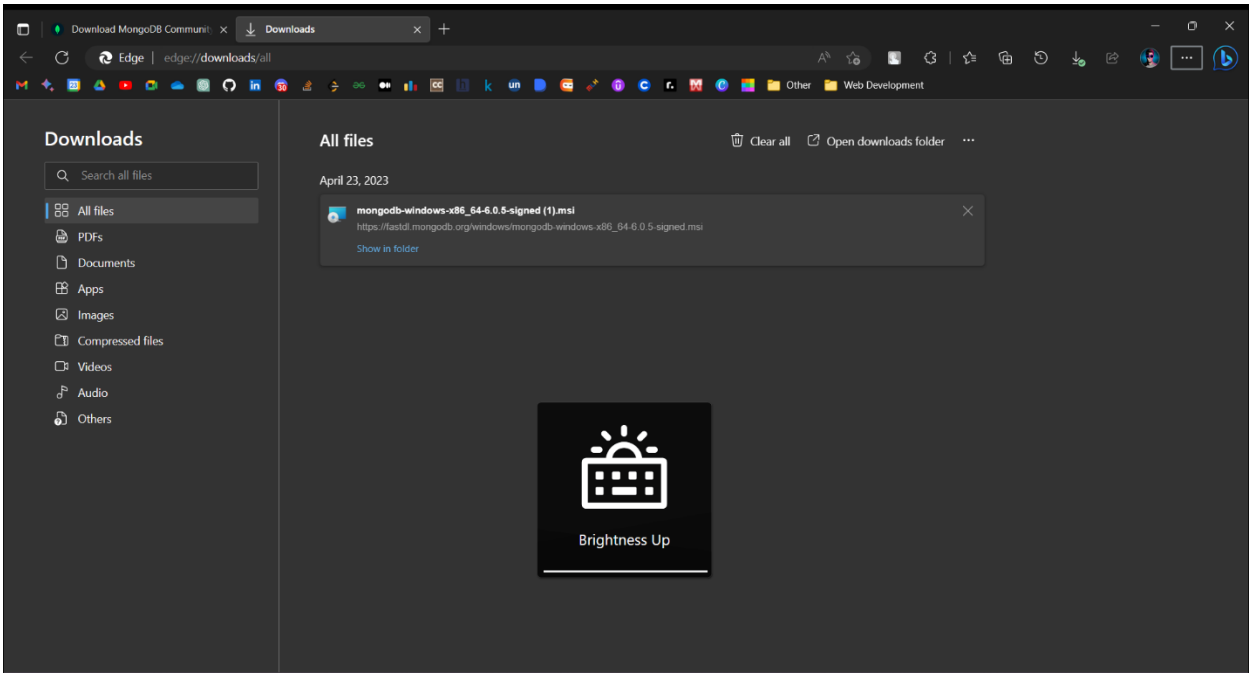
RDBMS
• Relational database • Need to design your tables, data structure, relations first. • Supports SQL query language • Table based • Row based • Column based • Each row will have same number of columns • Primary Key • Contains schema which is predefined

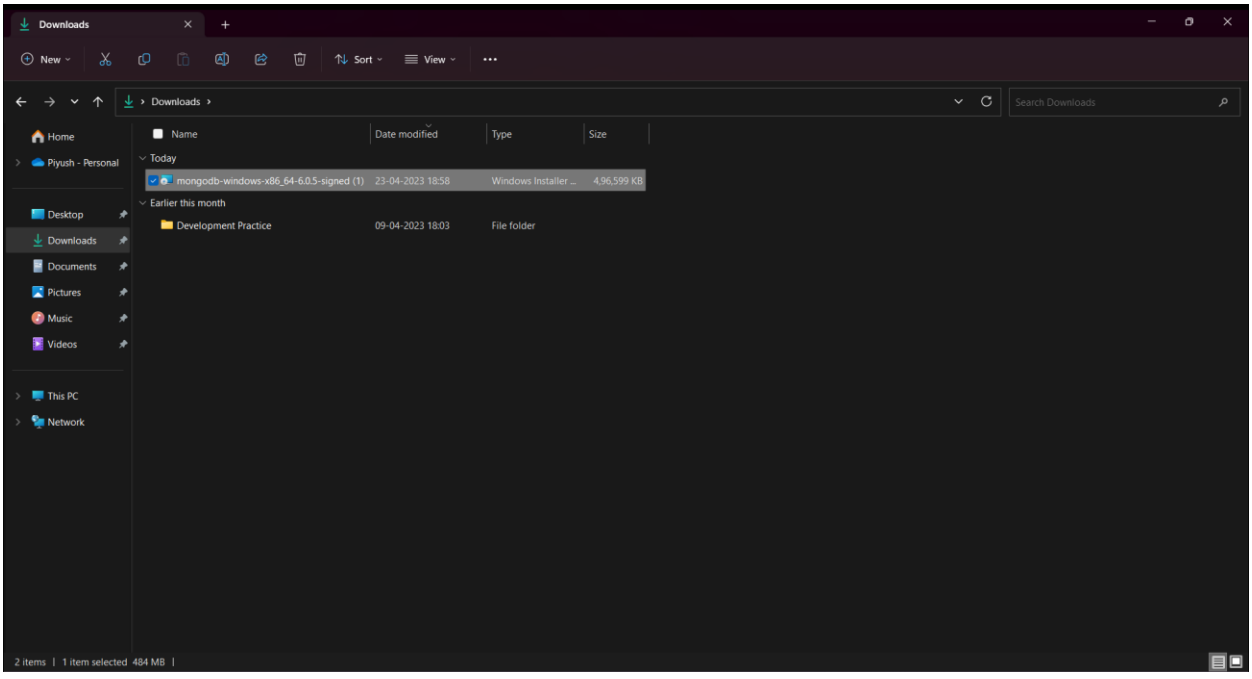
MongoDB
• Non-relational and document-oriented database • You can start coding without worrying about tables and modify your objects at a lesser cost of development. • Supports JSON query language • Collection based and key-value pair • Document based • Field based • Each document can have different number of fields • Primary Key (Default key **_id** provided by MongoDB itself) • Contains dynamic schema

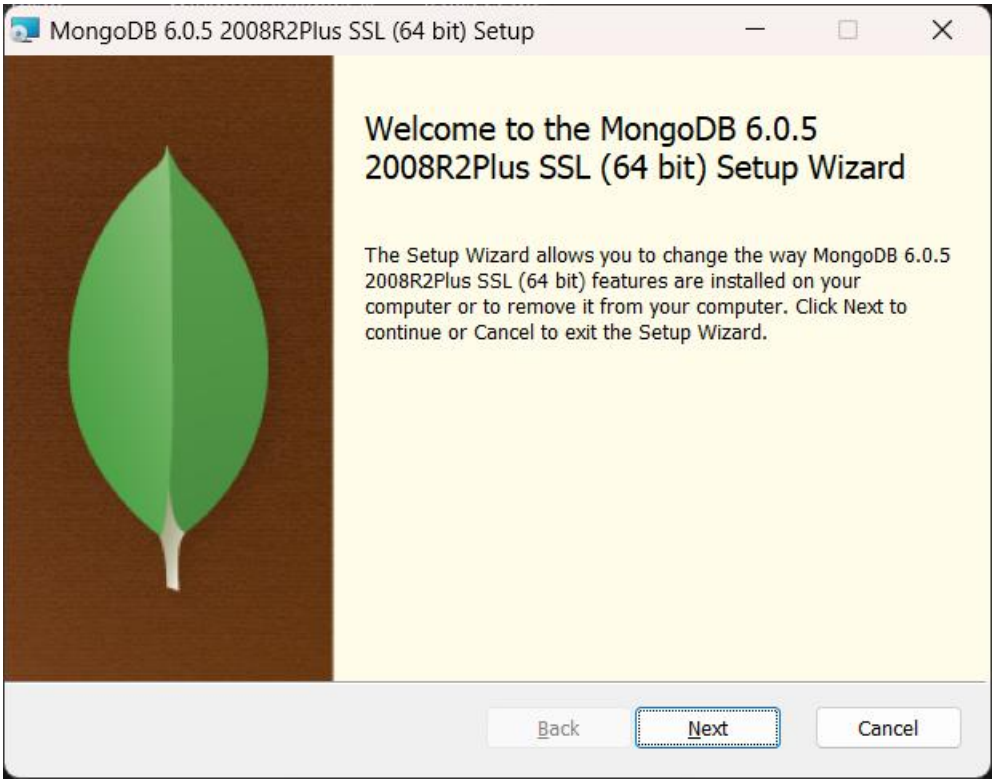
Output: Screenshots

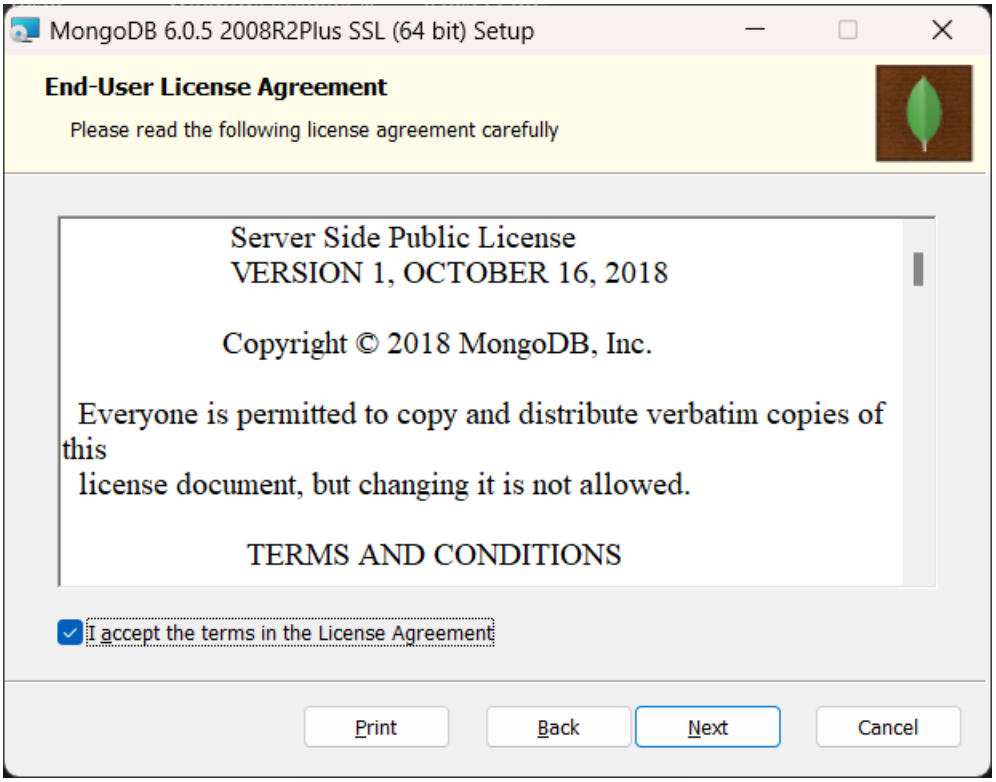


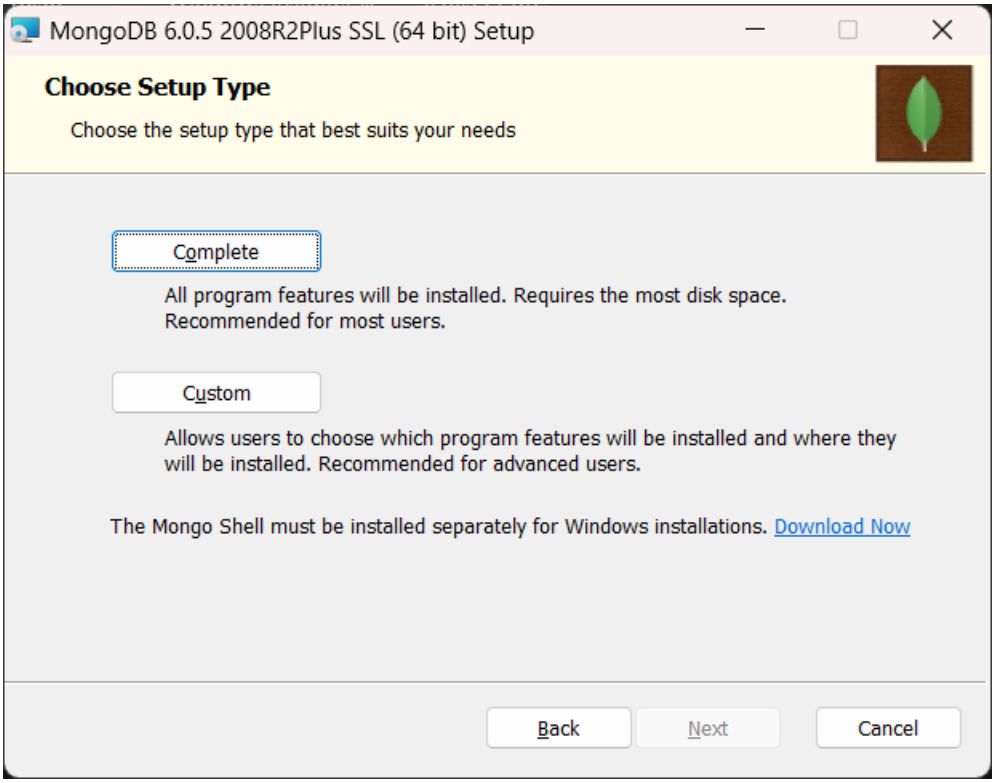


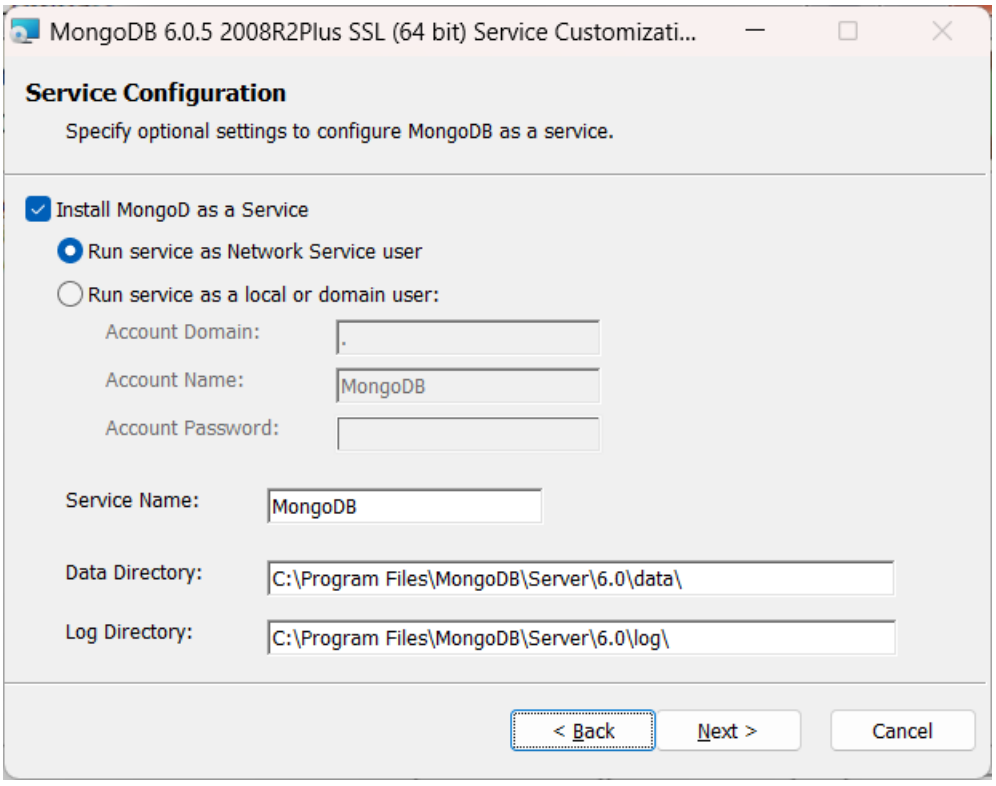

Download MongoDB Communit
Downloads
Edge | edge://downloads/all
Other
Web Development
Downloads
Search all files
All files
PDFs
Documents
Apps
Images
Compressed files
Videos
Audio
Others
All files
Clear all
Open downloads folder
April 23, 2023
mongodb-windows-x86_64-6.0.5-signed (1).msi
https://fastdl.mongodb.org/windows/mongodb-windows-x86_64-6.0.5-signed.msi
Show in folder
Brightness Up

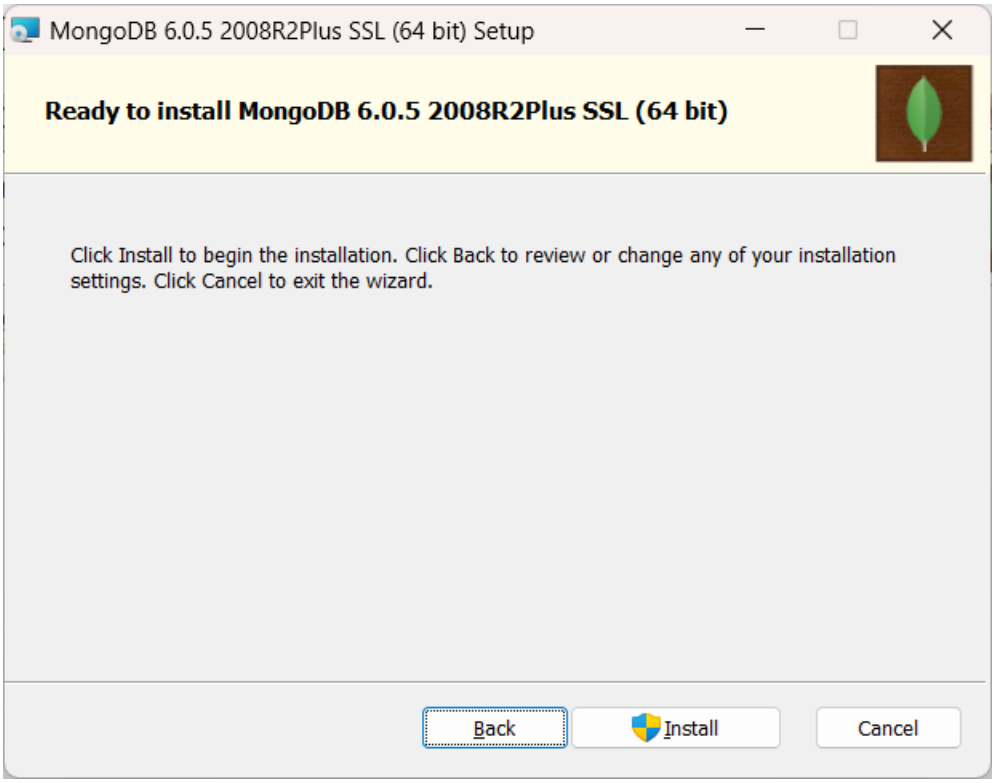

Downloads
New
Sort
View
Downloads
Search Downloads
Home
Piyush - Personal
Desktop
Downloads
Documents
Pictures
Music
Videos
This PC
Network
Name
Date modified
Type
Size
Today
mongodb-windows-x86_64-6.0.5-signed (1) 23-04-2023 18:58 Windows Installer ... 4,96,599 KB
Earlier this month
Development Practice 09-04-2023 18:03 File folder
2 items | 1 item selected 484 MB

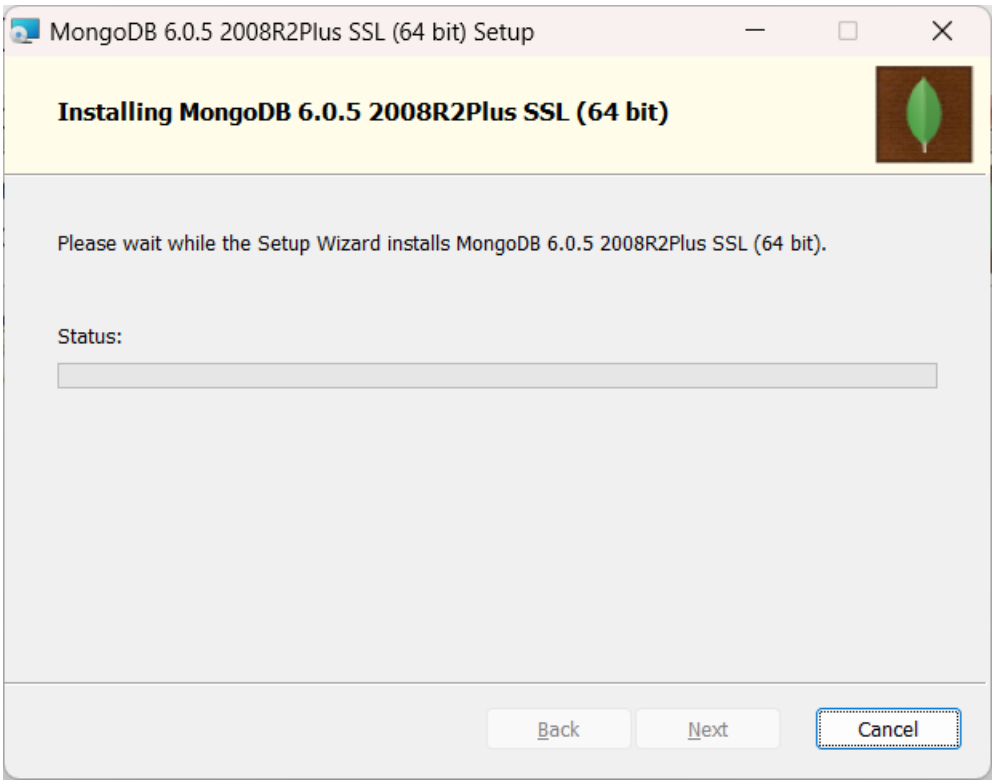

MongoDB 6.0.5 2008R2Plus SSL (64 bit) Setup
Welcome to the MongoDB 6.0.5 2008R2Plus SSL (64 bit) Setup Wizard
The Setup Wizard allows you to change the way MongoDB 6.0.5 2008R2Plus SSL (64 bit) features are installed on your computer or to remove it from your computer. Click Next to continue or Cancel to exit the Setup Wizard.
Back
Next
Cancel

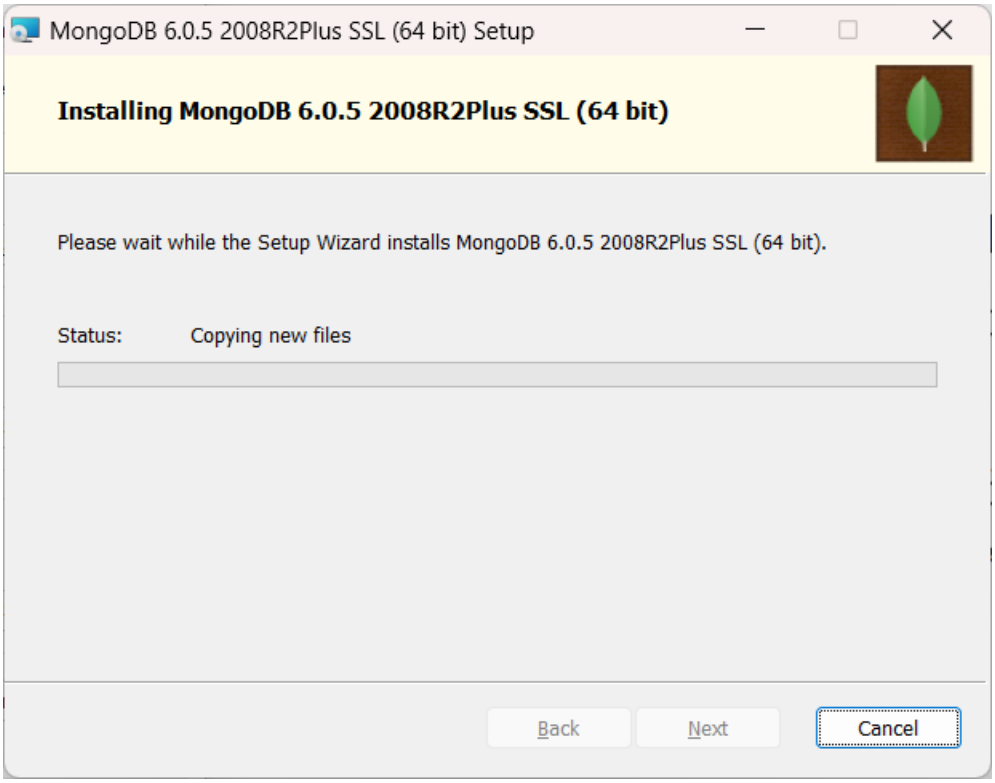

MongoDB 6.0.5 2008R2Plus SSL (64 bit) Setup
End-User License Agreement
Please read the following license agreement carefully
Server Side Public License
VERSION 1, OCTOBER 16, 2018
Copyright © 2018 MongoDB, Inc.
Everyone is permitted to copy and distribute verbatim copies of this
license document, but changing it is not allowed.
TERMS AND CONDITIONS
I accept the terms in the License Agreement
Print
Back
Next
Cancel

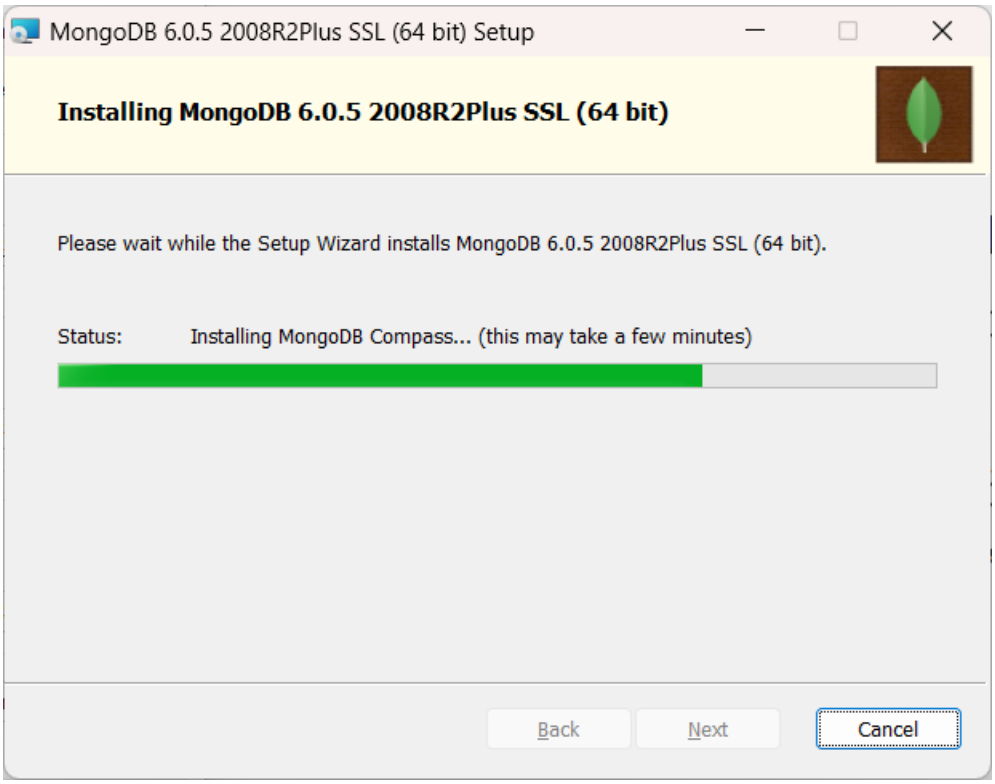

MongoDB 6.0.5 2008R2Plus SSL (64 bit) Setup
Choose Setup Type
Choose the setup type that best suits your needs
Complete
All program features will be installed. Requires the most disk space.
Recommended for most users.
Custom
Allows users to choose which program features will be installed and where they will be installed. Recommended for advanced users.
The Mongo Shell must be installed separately for Windows installations. Download Now
Back
Next
Cancel

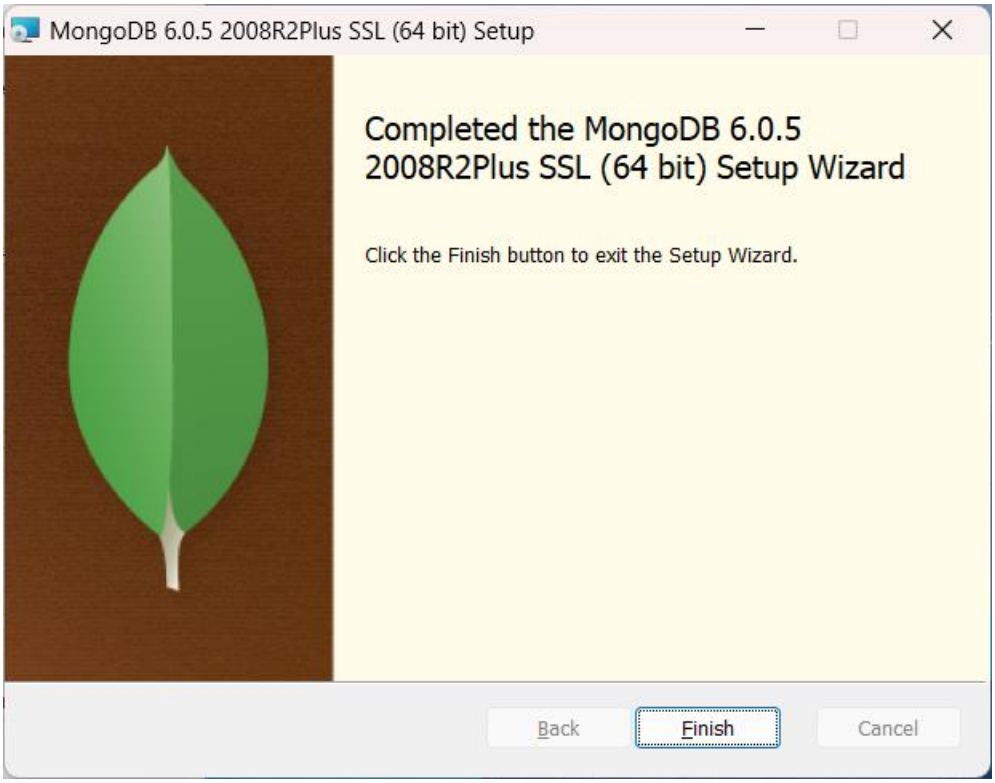

MongoDB 6.0.5 2008R2Plus SSL (64 bit) Service Customizati...
Service Configuration
Specify optional settings to configure MongoDB as a service.
Install MongoD as a Service
Run service as Network Service user
Run service as a local or domain user:
Account Domain:
.
Account Name:
MongoDB
Account Password:
Service Name:
MongoDB
Data Directory:
C:\Program Files\MongoDB\Server\6.0\data\
Log Directory:
C:\Program Files\MongoDB\Server\6.0\log\
< Back
Next >
Cancel

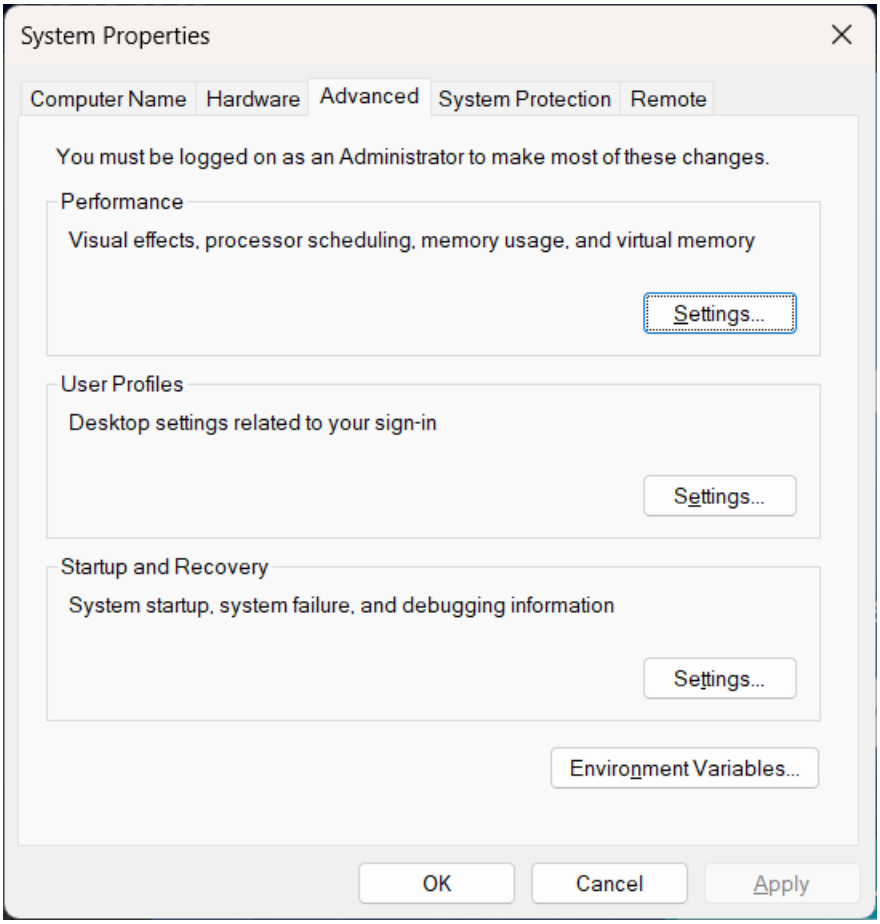

MongoDB 6.0.5 2008R2Plus SSL (64 bit) Setup
Ready to install MongoDB 6.0.5 2008R2Plus SSL (64 bit)
Click Install to begin the installation. Click Back to review or change any of your installation settings. Click Cancel to exit the wizard.
Back
Install
Cancel

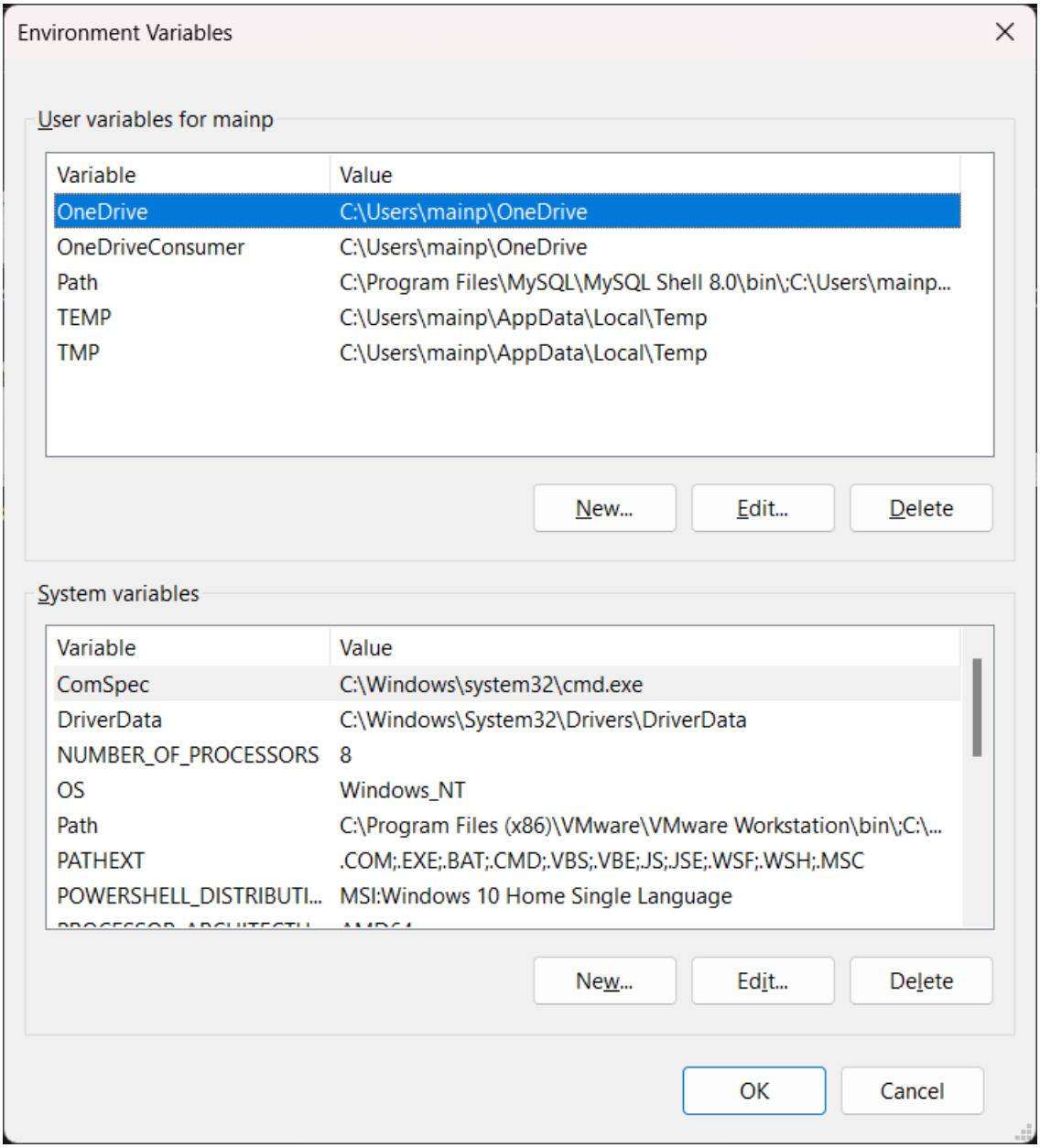

MongoDB 6.0.5 2008R2Plus SSL (64 bit) Setup
Installing MongoDB 6.0.5 2008R2Plus SSL (64 bit)
Please wait while the Setup Wizard installs MongoDB 6.0.5 2008R2Plus SSL (64 bit).
Status:
Back
Next
Cancel

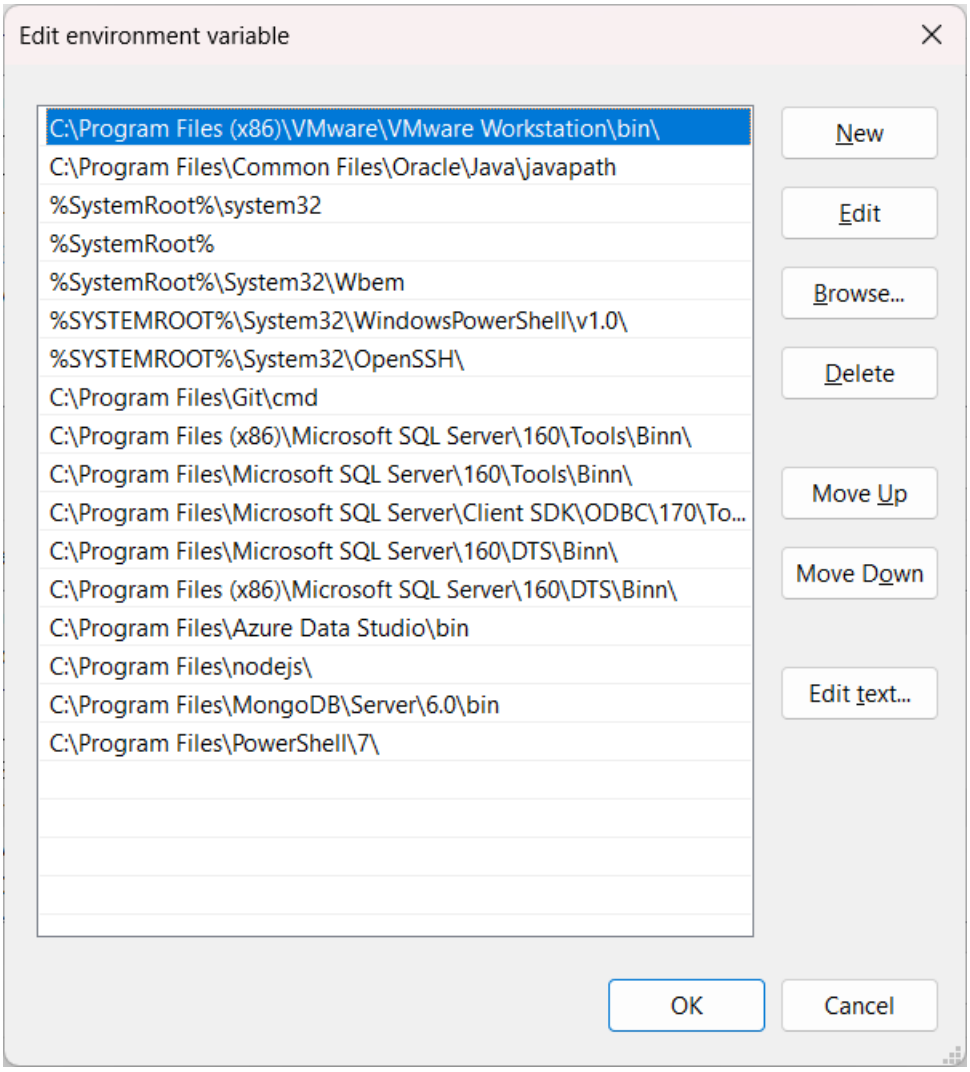

MongoDB 6.0.5 2008R2Plus SSL (64 bit) Setup
Installing MongoDB 6.0.5 2008R2Plus SSL (64 bit)
Please wait while the Setup Wizard installs MongoDB 6.0.5 2008R2Plus SSL (64 bit).
Status: Copying new files
Back
Next
Cancel

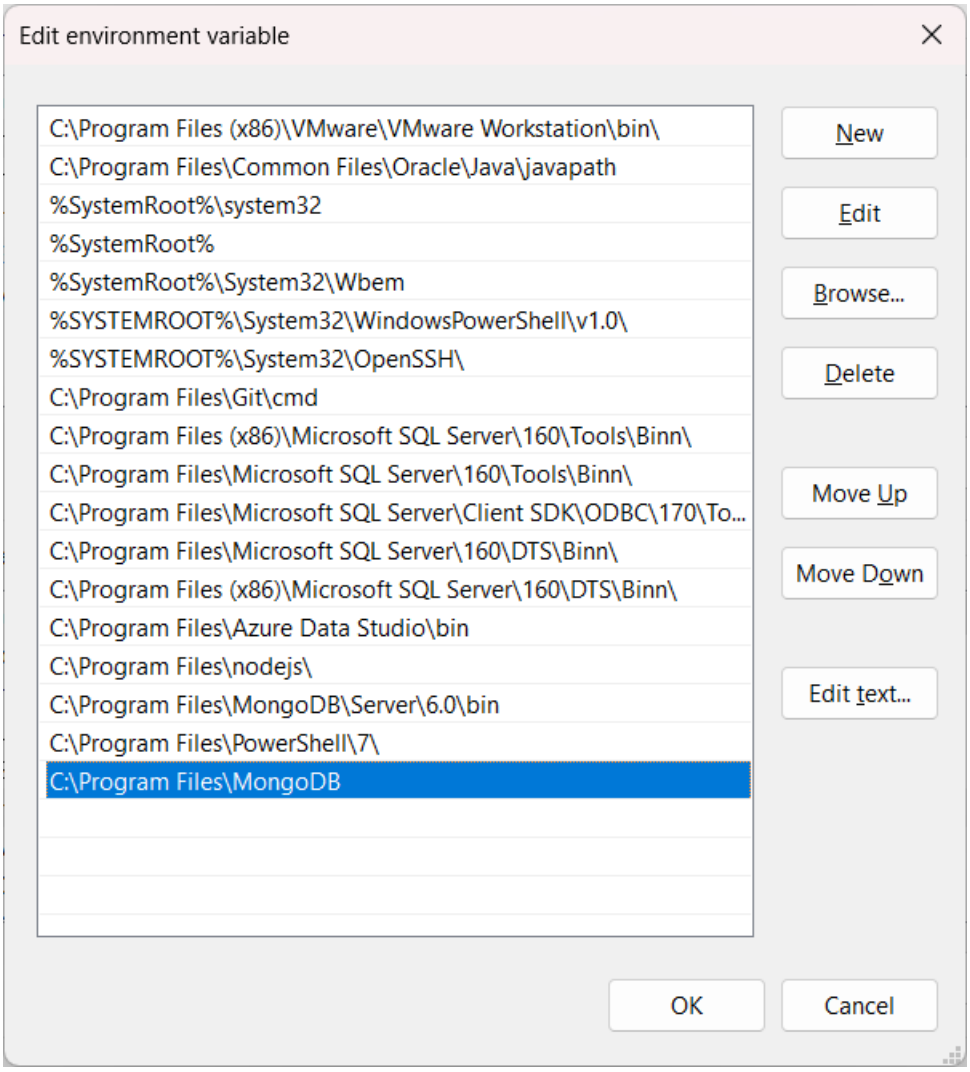

MongoDB 6.0.5 2008R2Plus SSL (64 bit) Setup
Installing MongoDB 6.0.5 2008R2Plus SSL (64 bit)
Please wait while the Setup Wizard installs MongoDB 6.0.5 2008R2Plus SSL (64 bit).
Status: Installing MongoDB Compass... (this may take a few minutes)
Back
Next
Cancel


MongoDB 6.0.5 2008R2Plus SSL (64 bit) Setup
Completed the MongoDB 6.0.5 2008R2Plus SSL (64 bit) Setup Wizard
Click the Finish button to exit the Setup Wizard.
Back
Finish
Cancel


System Properties
Computer Name
Hardware
Advanced
System Protection
Remote
You must be logged on as an Administrator to make most of these changes.
Performance
Visual effects, processor scheduling, memory usage, and virtual memory
Settings...
User Profiles
Desktop settings related to your sign-in
Settings...
Startup and Recovery
System startup, system failure, and debugging information
Settings...
Environment Variables...
OK
Cancel
Apply


Environment Variables
User variables for mainp
Variable
Value
OneDrive
C:\Users\mainp\OneDrive
OneDriveConsumer
C:\Users\mainp\OneDrive
Path
C:\Program Files\MySQL\MySQL Shell 8.0\bin\;C:\Users\mainp...
TEMP
C:\Users\mainp\AppData\Local\Temp
TMP
C:\Users\mainp\AppData\Local\Temp
New...
Edit...
Delete
System variables
Variable
Value
ComSpec
C:\Windows\system32\cmd.exe
DriverData
C:\Windows\System32\Drivers\DriverData
NUMBER_OF_PROCESSORS
8
OS
Windows_NT
Path
C:\Program Files (x86)\VMware\VMware Workstation\bin\;C:\...
PATHEXT
.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC
POWERSHELL_DISTRIBUTI...
MSI:Windows 10 Home Single Language
New...
Edit...
Delete
OK
Cancel


Edit environment variable
C:\Program Files (x86)\VMware\VMware Workstation\bin\
C:\Program Files\Common Files\Oracle\Java\javapath
%SystemRoot%\system32
%SystemRoot%
%SystemRoot%\System32\Wbem
%SYSTEMROOT%\System32\WindowsPowerShell\v1.0\
%SYSTEMROOT%\System32\OpenSSH\
C:\Program Files\Git\cmd
C:\Program Files (x86)\Microsoft SQL Server\160\Tools\Binn\
C:\Program Files\Microsoft SQL Server\160\Tools\Binn\
C:\Program Files\Microsoft SQL Server\Client SDK\ODBC\170\To...
C:\Program Files\Microsoft SQL Server\160\DTS\Binn\
C:\Program Files (x86)\Microsoft SQL Server\160\DTS\Binn\
C:\Program Files\Azure Data Studio\bin
C:\Program Files\nodejs\
C:\Program Files\MongoDB\Server\6.0\bin
C:\Program Files\PowerShell\7\
New
Edit
Browse...
Delete
Move Up
Move Down
Edit text...
OK
Cancel


Edit environment variable
C:\Program Files (x86)\VMware\VMware Workstation\bin\
C:\Program Files\Common Files\Oracle\Java\javapath
%SystemRoot%\system32
%SystemRoot%
%SystemRoot%\System32\Wbem
%SYSTEMROOT%\System32\WindowsPowerShell\v1.0\
%SYSTEMROOT%\System32\OpenSSH\
C:\Program Files\Git\cmd
C:\Program Files (x86)\Microsoft SQL Server\160\Tools\Binn\
C:\Program Files\Microsoft SQL Server\160\Tools\Binn\
C:\Program Files\Microsoft SQL Server\Client SDK\ODBC\170\To...
C:\Program Files\Microsoft SQL Server\160\DTS\Binn\
C:\Program Files (x86)\Microsoft SQL Server\160\DTS\Binn\
C:\Program Files\Azure Data Studio\bin
C:\Program Files\nodejs\
C:\Program Files\MongoDB\Server\6.0\bin
C:\Program Files\PowerShell\7\
C:\Program Files\MongoDB
New
Edit
Browse...
Delete
Move Up
Move Down
Edit text...
OK
Cancel

Preparatory Questions

Q1) Point out the correct statement:

1) A database is a set of key-value pairs
2) A MongoDB deployment hosts a number of databases
3) A document holds a set of collections
4) All of the mentioned

Q2) MongoDB stores all documents in:

1) tables
2) collections
3) rows
4) all the mentioned

Q3) BSON is a binary representation of _____ documents,

1) JSON
2) XML
3) JScript
4) All of the mentioned

Q4) The maximum size of a MongoDB document is

1) 2 MB
2) 16 MB
3) 12 MB
4) There is no maximum size. It depends on RAM

Q5) Which of the following statements is true?

1) MongoDB cannot be used as a file system.
2) MongoDB can run over single servers only.
3) Embedded documents and arrays reduce need for joins.
4) None

Experiment 9

Student Name and Roll Number: Piyush Gambhir – 21CSU349
Semester /Section: Semester-IV – AIML-B (AL-3)
Link to Code: Piyush-Gambhir/NCU-CSL214-DBMS-Lab_Manual (github.com)
Date: 17.04.2023
Faculty Signature:
Marks:

Objective

Create COMPANY database using NoSQL database - MongoDB.

Program Outcome

- The students will be able to perform basic CRUD operations in MongoDB.

Problem Statement

1) Create a COMPANY database with the following collections:
   - Employee with fields Emp ID, Ename, Age, Mobile, Email, Address, Dno
   - Department with fields Dnumber, Dname, Dlocation
   - Project with fields Pname, Pnumber, Plocation, ControlDept

2) Insert 5 documents in each collection using different versions of "Insert" command.

3) Select all documents in a collection.

4) Retrieve the details of all projects that are being run in department number 50.

5) Retrieve the details of the first project that is being run in department number 50 in formatted manner.

6) Return the formatted/structured project details of all projects in department number "5" and project name as "ProductZ".

7) Return the project details (including project name, project number and department number and excluding project location and default primary key) of all projects with department number "5" or project name as "ProductZ".

8) Return the formatted/structured project details of all projects with department number less than "10" and project name as "ProductZ" or project location as "Delhi".

Background Study

1) **use:** This command is used to create a database in MongoDB. It will return an existing database or will create a new database if it dosen't exist.

2) **createCollection( ):** To create a collections in a database

3) MongoDB provides the following methods to insert documents into a collection:

4) **db.collection.insert():** used to insert one or multiple documents. To insert multiple documents in a single query, pass an array of documents in insert() command.

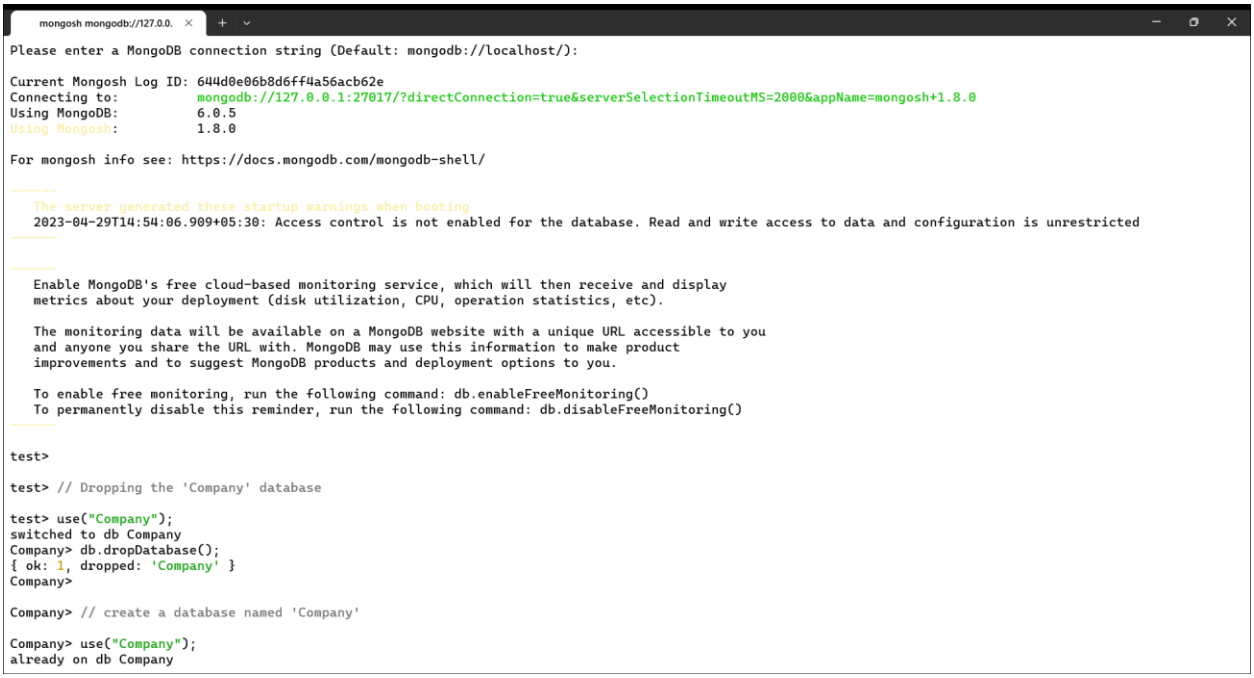
5) **db.collection.insertOne():** used to insert only one document.

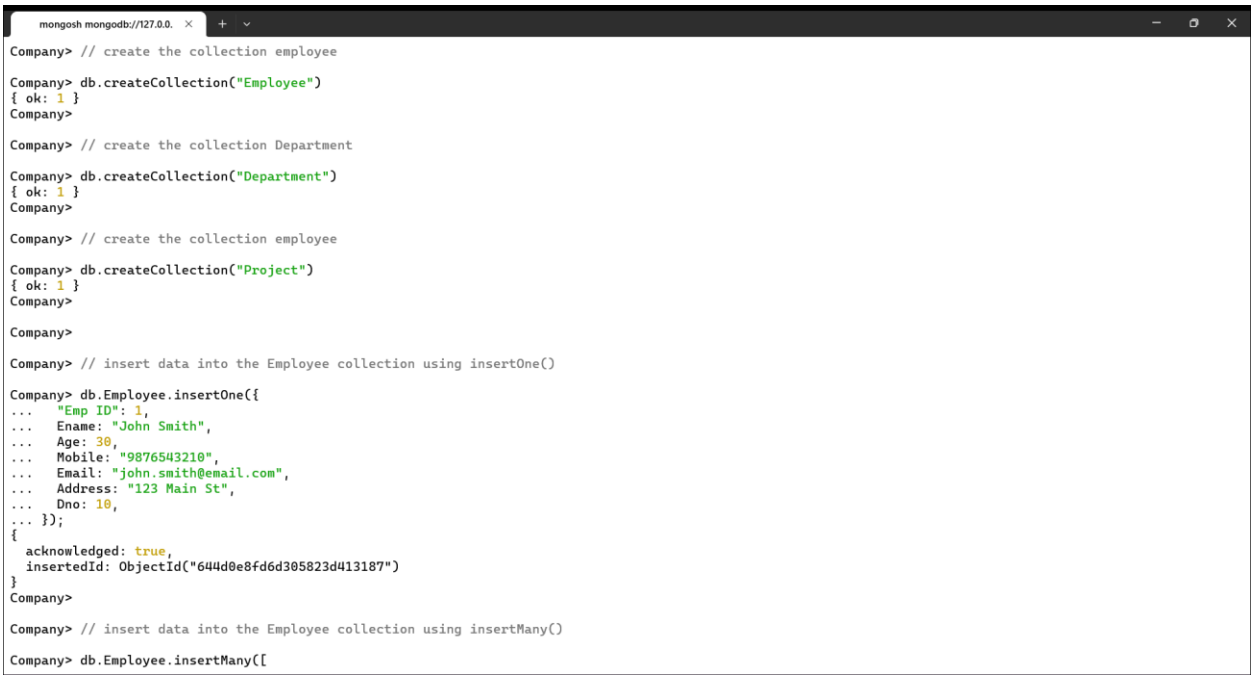
6) **db.collection.insertMany():** used to insert multiple documents. To insert multiple documents in a single query, pass an array of documents in insert() command.

7) **db.collection.find():** this method is provided in MongoDB to read documents from a collection. It returns a **cursor** to the matching documents. db.collection.find(<query filter>, <projection>) accepts second optional parameter that is list of fields that you want to retrieve in the form { field1: <value>, field2: <value> ... }.

8) **db.collection.findOne()**: it returns the first occurrence of document, otherwise null.

Output: Screenshots





```
mongosh mongodb://127.0.0.
Company> // insert data into the Employee collection using insertMany()

Company> db.Employee.insertMany([
...   {
...     "Emp ID": 2,
...     Ename: "Jane Doe",
...     Age: 25,
...     Mobile: "9876543211",
...     Email: "jane.doe@email.com",
...     Address: "456 1st St",
...     Dno: 2,
...   },
...   {
...     "Emp ID": 3,
...     Ename: "Bob Johnson",
...     Age: 45,
...     Mobile: "9876543212",
...     Email: "bob.johnson@email.com",
...     Address: "789 2nd St",
...     Dno: 3,
...   },
...   {
...     "Emp ID": 4,
...     Ename: "Sarah Lee",
...     Age: 35,
...     Mobile: "9876543213",
...     Email: "sarah.lee@email.com",
...     Address: "1010 3rd St",
...     Dno: 4,
...   },
...   {
...     "Emp ID": 5,
...     Ename: "Tommy Nguyen",
...     Age: 28,
...     Mobile: "9876543214",
...     Email: "tommy.nguyen@email.com",
...     Address: "1111 4th St",
...     Dno: 5,
...   },
```

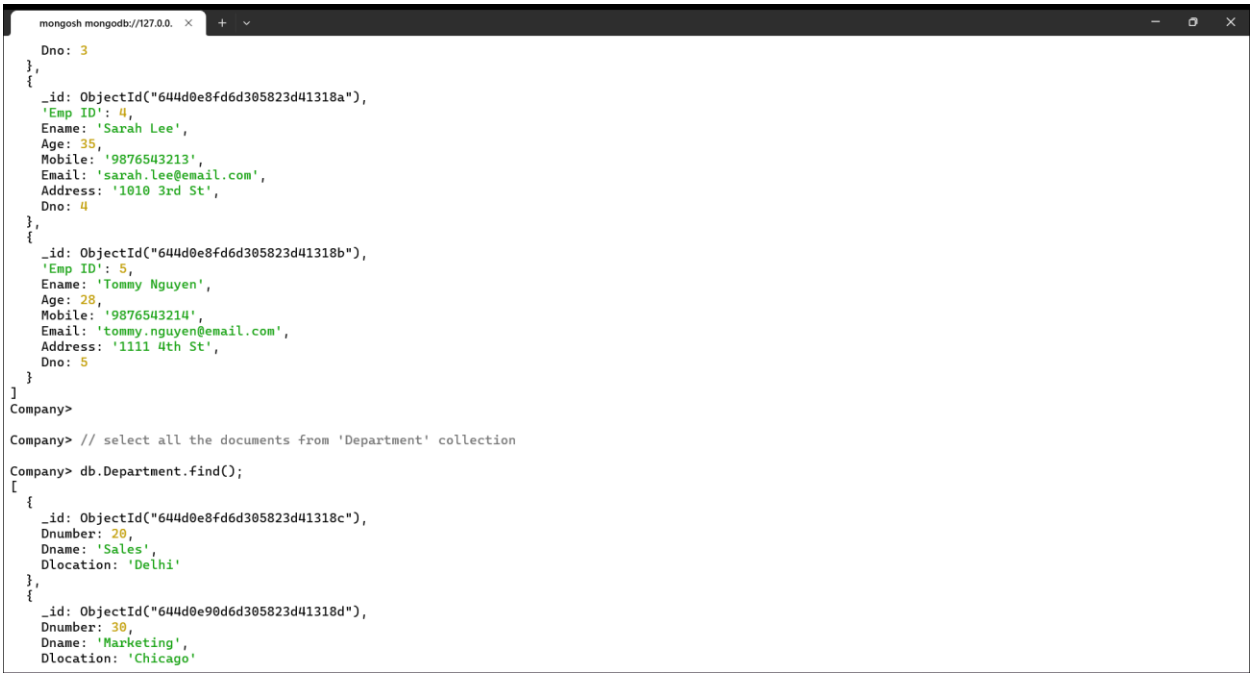
```
mongosh mongodb://127.0.0.
...     Ename: "Sarah Lee",
...     Age: 35,
...     Mobile: "9876543213",
...     Email: "sarah.lee@email.com",
...     Address: "1010 3rd St",
...     Dno: 4,
...   },
...   {
...     "Emp ID": 5,
...     Ename: "Tommy Nguyen",
...     Age: 28,
...     Mobile: "9876543214",
...     Email: "tommy.nguyen@email.com",
...     Address: "1111 4th St",
...     Dno: 5,
...   },
... ]);
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId("644d0e8fd6d305823d413188"),
    '1': ObjectId("644d0e8fd6d305823d413189"),
    '2': ObjectId("644d0e8fd6d305823d41318a"),
    '3': ObjectId("644d0e8fd6d305823d41318b")
  }
}
Company>

Company> // insert data into the Department collection using insertOne()

Company> db.Department.insertOne({
...   Dnumber: 20,
...   Dname: "Sales",
...   Dlocation: "Delhi",
... });
{
  acknowledged: true,
  insertedId: ObjectId("644d0e8fd6d305823d41318c")
}
Company>
```

```
mongosh mongodb://127.0.0.
Company>

Company> // insert data into the Department collection using insertMany()

Company> db.Department.insertMany([
...   {
...     Dnumber: 30,
...     Dname: "Marketing",
...     Dlocation: "Chicago",
...   },
...   {
...     Dnumber: 40,
...     Dname: "Human Resources",
...     Dlocation: "Houston",
...   },
...   {
...     Dnumber: 50,
...     Dname: "Finance",
...     Dlocation: "Delhi",
...   },
...   {
...     Dnumber: 60,
...     Dname: "IT",
...     Dlocation: "New York",
...   },
... ]);
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId("644d0e90d6d305823d41318d"),
    '1': ObjectId("644d0e90d6d305823d41318e"),
    '2': ObjectId("644d0e90d6d305823d41318f"),
    '3': ObjectId("644d0e90d6d305823d413190")
  }
}
Company>

Company> // insert data into the Project collection using insertOne()

Company> db.Project.insertOne({
```

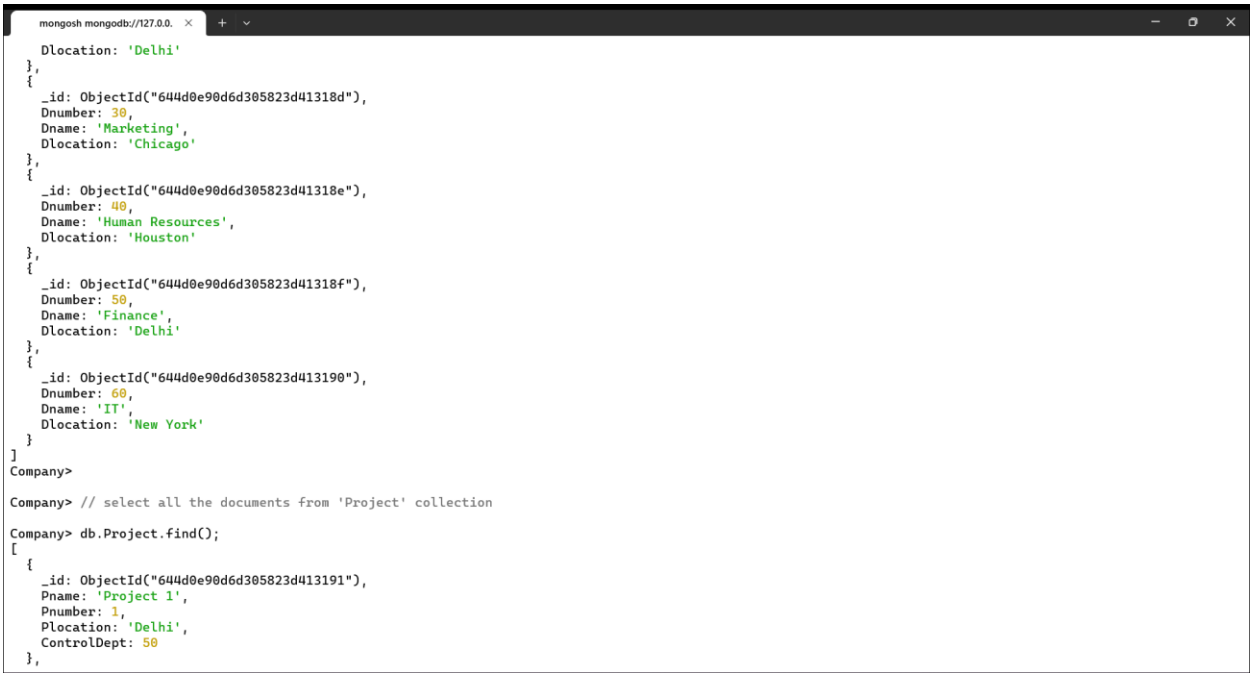
```
mongosh mongodb://127.0.0.

Company> // insert data into the Project collection using insertOne()

Company> db.Project.insertOne({
...   Pname: "Project 1",
...   Pnumber: 1,
...   Plocation: "Delhi",
...   ControlDept: 50,
... });
{
  acknowledged: true,
  insertedId: ObjectId("644d0e90d6d305823d413191")
}
Company>

Company> // insert data into the Project collection using insertMany()

Company> db.Project.insertMany([
...   {
...     Pname: "Project 2",
...     Pnumber: 2,
...     Plocation: "Chicago",
...     ControlDept: 2,
...   },
...   {
...     Pname: "Project 3",
...     Pnumber: 3,
...     Plocation: "Houston",
...     ControlDept: 50,
...   },
...   {
...     Pname: "Project 4",
...     Pnumber: 4,
...     Plocation: "Delhi",
...     ControlDept: 5,
...   },
...   {
...     Pname: "ProductZ",
...     Pnumber: 5,
...     Plocation: "New York",
```

```
mongosh mongodb://127.0.0.
}
Company>

Company> // insert data into the Project collection using insertMany()

Company> db.Project.insertMany([
...   {
...     Pname: "Project 2",
...     Pnumber: 2,
...     Plocation: "Chicago",
...     ControlDept: 2,
...   },
...   {
...     Pname: "Project 3",
...     Pnumber: 3,
...     Plocation: "Houston",
...     ControlDept: 50,
...   },
...   {
...     Pname: "Project 4",
...     Pnumber: 4,
...     Plocation: "Delhi",
...     ControlDept: 5,
...   },
...   {
...     Pname: "ProductZ",
...     Pnumber: 5,
...     Plocation: "New York",
...     ControlDept: 5,
...   },
... ]);
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId("644d0e90d6d305823d413192"),
    '1': ObjectId("644d0e90d6d305823d413193"),
    '2': ObjectId("644d0e90d6d305823d413194"),
    '3': ObjectId("644d0e90d6d305823d413195")
  }
}
```

```
mongosh mongodb://127.0.0.

Company> // Select all documents from the 'Employee' collection

Company> db.Employee.find();
[
  {
    _id: ObjectId("644d0e8fd6d305823d413187"),
    'Emp ID': 1,
    Ename: 'John Smith',
    Age: 30,
    Mobile: '9876543210',
    Email: 'john.smith@email.com',
    Address: '123 Main St',
    Dno: 10
  },
  {
    _id: ObjectId("644d0e8fd6d305823d413188"),
    'Emp ID': 2,
    Ename: 'Jane Doe',
    Age: 25,
    Mobile: '9876543211',
    Email: 'jane.doe@email.com',
    Address: '456 1st St',
    Dno: 2
  },
  {
    _id: ObjectId("644d0e8fd6d305823d413189"),
    'Emp ID': 3,
    Ename: 'Bob Johnson',
    Age: 45,
    Mobile: '9876543212',
    Email: 'bob.johnson@email.com',
    Address: '789 2nd St',
    Dno: 3
  },
  {
    _id: ObjectId("644d0e8fd6d305823d41318a"),
    'Emp ID': 4,
    Ename: 'Sarah Lee',
    Age: 35,
```



```
mongosh mongodb://127.0.0.
    Dno: 3
  },
  {
    _id: ObjectId("644d0e8fd6d305823d41318a"),
    'Emp ID': 4,
    Ename: 'Sarah Lee',
    Age: 35,
    Mobile: '9876543213',
    Email: 'sarah.lee@email.com',
    Address: '1010 3rd St',
    Dno: 4
  },
  {
    _id: ObjectId("644d0e8fd6d305823d41318b"),
    'Emp ID': 5,
    Ename: 'Tommy Nguyen',
    Age: 28,
    Mobile: '9876543214',
    Email: 'tommy.nguyen@email.com',
    Address: '1111 4th St',
    Dno: 5
  }
]
Company>

Company> // select all the documents from 'Department' collection

Company> db.Department.find();
[
  {
    _id: ObjectId("644d0e8fd6d305823d41318c"),
    Dnumber: 20,
    Dname: 'Sales',
    Dlocation: 'Delhi'
  },
  {
    _id: ObjectId("644d0e90d6d305823d41318d"),
    Dnumber: 30,
    Dname: 'Marketing',
    Dlocation: 'Chicago'
```



```
mongosh mongodb://127.0.0.
    Dlocation: 'Delhi'
  },
  {
    _id: ObjectId("644d0e90d6d305823d41318d"),
    Dnumber: 30,
    Dname: 'Marketing',
    Dlocation: 'Chicago'
  },
  {
    _id: ObjectId("644d0e90d6d305823d41318e"),
    Dnumber: 40,
    Dname: 'Human Resources',
    Dlocation: 'Houston'
  },
  {
    _id: ObjectId("644d0e90d6d305823d41318f"),
    Dnumber: 50,
    Dname: 'Finance',
    Dlocation: 'Delhi'
  },
  {
    _id: ObjectId("644d0e90d6d305823d413190"),
    Dnumber: 60,
    Dname: 'IT',
    Dlocation: 'New York'
  }
]
Company>

Company> // select all the documents from 'Project' collection

Company> db.Project.find();
[
  {
    _id: ObjectId("644d0e90d6d305823d413191"),
    Pname: 'Project 1',
    Pnumber: 1,
    Plocation: 'Delhi',
    ControlDept: 50
  },
```

```
mongosh mongodb://127.0.0.
Company>

Company> // select all the documents from 'Project' collection

Company> db.Project.find();
[
  {
    _id: ObjectId("644d0e90d6d305823d413191"),
    Pname: 'Project 1',
    Pnumber: 1,
    Plocation: 'Delhi',
    ControlDept: 50
  },
  {
    _id: ObjectId("644d0e90d6d305823d413192"),
    Pname: 'Project 2',
    Pnumber: 2,
    Plocation: 'Chicago',
    ControlDept: 2
  },
  {
    _id: ObjectId("644d0e90d6d305823d413193"),
    Pname: 'Project 3',
    Pnumber: 3,
    Plocation: 'Houston',
    ControlDept: 50
  },
  {
    _id: ObjectId("644d0e90d6d305823d413194"),
    Pname: 'Project 4',
    Pnumber: 4,
    Plocation: 'Delhi',
    ControlDept: 5
  },
  {
    _id: ObjectId("644d0e90d6d305823d413195"),
    Pname: 'ProductZ',
    Pnumber: 5,
    Plocation: 'New York',
    ControlDept: 5
```

```
mongosh mongodb://127.0.0.
  },
  {
    _id: ObjectId("644d0e90d6d305823d413195"),
    Pname: 'ProductZ',
    Pnumber: 5,
    Plocation: 'New York',
    ControlDept: 5
  }
]
Company>

Company> // retrieve the details of all projects that are being run in department number 50

Company> db.Project.find({ ControlDept: 50 });
[
  {
    _id: ObjectId("644d0e90d6d305823d413191"),
    Pname: 'Project 1',
    Pnumber: 1,
    Plocation: 'Delhi',
    ControlDept: 50
  },
  {
    _id: ObjectId("644d0e90d6d305823d413193"),
    Pname: 'Project 3',
    Pnumber: 3,
    Plocation: 'Houston',
    ControlDept: 50
  }
]
Company>

Company> // retrieve the details of the first project that is being run in department number 50 in formatted manner

Company> db.Project.find({ ControlDept: 50 }).pretty();
[
  {
    _id: ObjectId("644d0e90d6d305823d413191"),
    Pname: 'Project 1',
    Pnumber: 1,
```

```
mongosh mongodb://127.0.0.
Company>

Company> // retrieve the details of the first project that is being run in department number 50 in formatted manner

Company> db.Project.find({ ControlDept: 50 }).pretty();
[
  {
    _id: ObjectId("644d0e90d6d305823d413191"),
    Pname: 'Project 1',
    Pnumber: 1,
    Plocation: 'Delhi',
    ControlDept: 50
  },
  {
    _id: ObjectId("644d0e90d6d305823d413193"),
    Pname: 'Project 3',
    Pnumber: 3,
    Plocation: 'Houston',
    ControlDept: 50
  }
]
Company>

Company> // return the formatted/structured project details of all projects in department number “5” and project name as “ProductZ”

Company> db.Project.find({ ControlDept: 5, Pname: "ProductZ" }).pretty();
[
  {
    _id: ObjectId("644d0e90d6d305823d413195"),
    Pname: 'ProductZ',
    Pnumber: 5,
    Plocation: 'New York',
    ControlDept: 5
  }
]
Company>

ult primary key) of all projects with department number “5” or project name as “ProductZ”ment number and excluding project location and defa

Company> db.Project.find(
```

```
mongosh mongodb://127.0.0.
  }
]
Company>

ult primary key) of all projects with department number “5” or project name as “ProductZ”ment number and excluding project location and defa

Company> db.Project.find(
...   { $or: [{ ControlDept: 5 }, { Pname: "ProductZ" }] },
...   { Pname: 1, Pnumber: 1, ControlDept: 1, _id: 0 }
... );
[
  { Pname: 'Project 4', Pnumber: 4, ControlDept: 5 },
  { Pname: 'ProductZ', Pnumber: 5, ControlDept: 5 }
]
Company>

ctZ” or project location as “Delhi”ructured project details of all projects with department number less than “10” and project name as “Produ

Company> db.Project.find({
...   $and: [
...     { ControlDept: { $lt: 10 } }, { $or: [{ Pname: "ProductZ" }, { Plocation: "Delhi" }] },
...   ],
... }).pretty();
[
  {
    _id: ObjectId("644d0e90d6d305823d413194"),
    Pname: 'Project 4',
    Pnumber: 4,
    Plocation: 'Delhi',
    ControlDept: 5
  },
  {
    _id: ObjectId("644d0e90d6d305823d413195"),
    Pname: 'ProductZ',
    Pnumber: 5,
    Plocation: 'New York',
    ControlDept: 5
  }
]
Company>
```

Preparatory Questions

Q1) Which of the following method is used to query documents in collections?

1) find
2) move
3) shell
4) replace

Q2) When you query a collection, MongoDB returns a _____ object that contains the results of the query.

1) row
2) cursor
3) colums
4) none of the mentioned

Q3) Which of the following method is called while accessing documents using the array index notation ?

1) cur.toArray()
2) cursor.toArray()
3) doc.toArray()
4) all of the mentioned

Q4) The mongo shell and the drivers provide several cursor methods that call on the cursor returned by the _____ method to modify its behavior.

1) cursor()
2) find()
3) findc()
4) none of the mentioned

Q5) Which of the following method corresponds to Order by clause in SQL?

1) sort()
2) order()
3) orderby()
4) all of the mentioned

Experiment 10

Student Name and Roll Number: Piyush Gambhir – 21CSU349
Semester /Section: Semester-IV – AIML-B (AL-3)
Link to Code: Piyush-Gambhir/NCU-CSL214-DBMS-Lab_Manual (github.com)
Date: 24.04.2023
Faculty Signature:
Marks:

Objective

Retrieve data from NoSQL database - MongoDB.

Program Outcome

- The students will be able to perform advanced CRUD operations in MongoDB.

Problem Statement

Create a database with the following collections:

Project

Possible Fields:

- Project Name (Pname)
- Project Number (Pnumber)
- Department Number in which project is being run (Dnum)
- Project Locations (Plocation)
  - Plocation can be an array of multiple locations (or)
  - Plocation can be a composite field with sub-fields – city, state, country

Department

Possible Fields:

- Department Number (Dno)
- Department Name (Dname)
- Department Manager (Dmanager)

Execute the following queries:

1) Insert multiple documents (2-3) together in the Project collection with atleast 1 project location as a composite attribute

2) Write the “Select * from Project” equivalent query in MongoDB.

3) Write a MongoDB query to update the project location of all projects to “Pune” and department number to “20” with project name “ProjectX”.

4) Write a MongoDB query to delete all the documents of collection “Project” with department number “4”.

5) Write a MongoDB query to retrieve the project details of all projects with project location as “Delhi” and “Mumbai” both assuming the following documents exist.

   { "_id" : ObjectId("5cc7fe36e37496f1a5e7e268"),"Pname" : "ProductK", "Pnumber" : 45,"Plocation" : ["Delhi","Mumbai","Pune"], "Dnum" : 3}
   { "_id" : ObjectId("5cc7fe36e37496f1a5e7e268"),"Pname" : "ProductT", "Pnumber" : 56,"Plocation" : ["Delhi","Mumbai"], "Dnum" : 15}
   { "_id" : ObjectId("5cc7fe36e37496f1a5e7e268"),"Pname" : "ProductU", "Pnumber" : 5,"Plocation" : "Delhi", "Dnum" : 6}
   { "_id" : ObjectId("5cc7fe36e37496f1a5e7e268"),"Pname" : "ProductL", "Pnumber" : 4,"Plocation" : "Mumbai", "Dnum" : 7}

6) Write a MongoDB query creating a One to Many Relation between a document of “Project” Collection and “Department” Collection.

7) Write a MongoDB query to retrieve the project details → project name, department number, department name and department manager of projects with department manager (for department in which the project is being run) as “ABC”.

Background Study

1) pretty() is used to display the query obtained in a neat manner with all the fields clearly depicted one below the other.

2) Data for a particular field can also be stored as an array with multiple values for the field.

3) $all is used to match the array items for the particular field.
   - *db.project.find({Plocation:{$all:["Delhi","Mumbai"]}}).pretty()*

4) $elemMatch: This operator matches documents that contain an array field with at least one element that matches all the specified query criteria.
   - Syntax: { <field>: { $elemMatch: { <query1>, <query2>, ... } } }

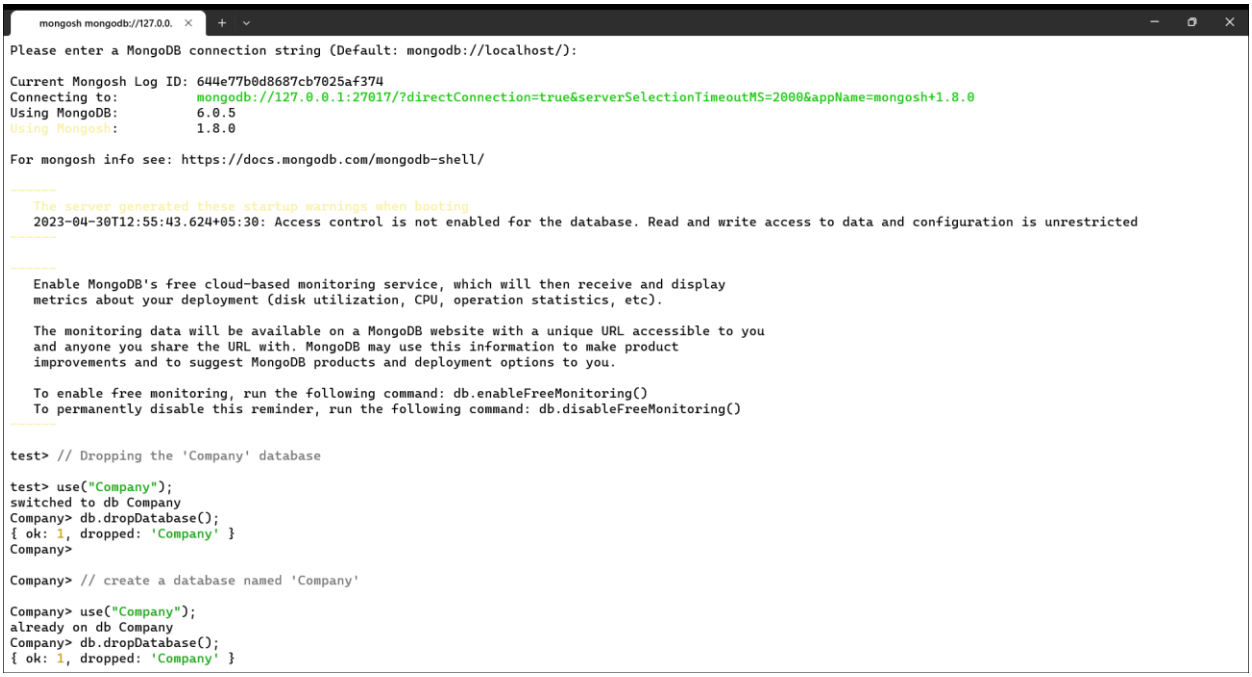
5) Fields can be embedded in the field too and while using find in such a case, always refer the field embedded in another field by using a dot notation and in quotes.

6) AND/OR Operations

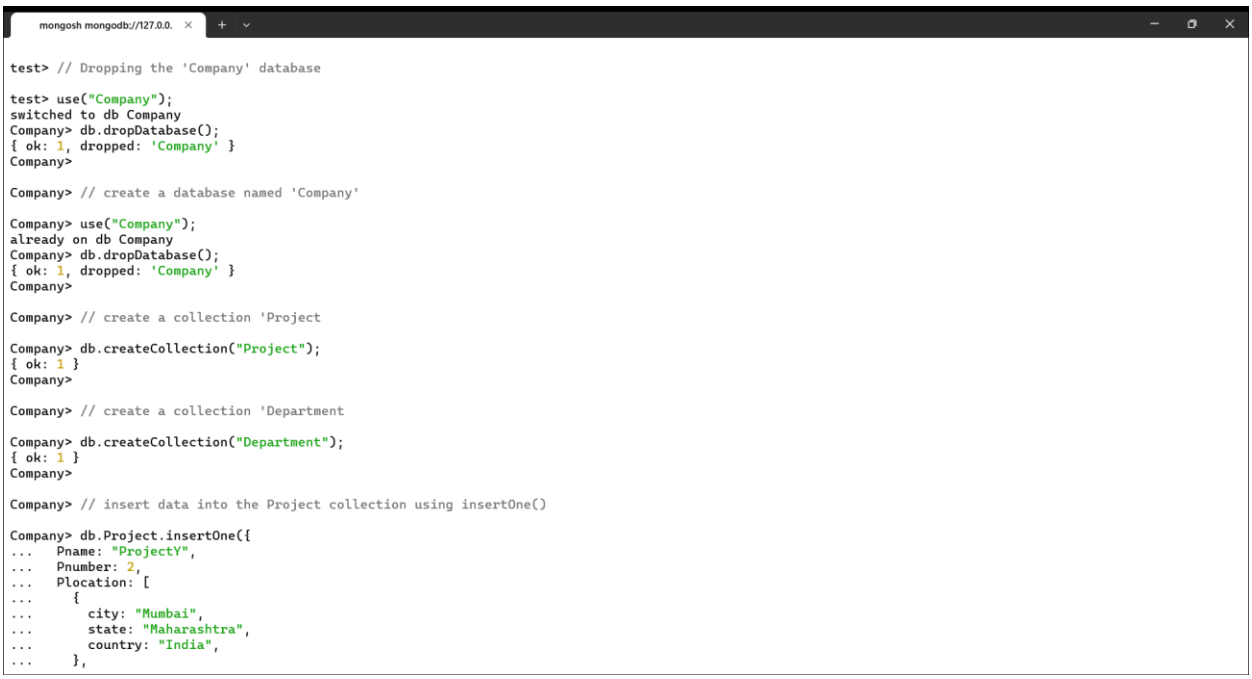
Operation	Syntax	Example
AND	>db.collection_name.find({$and: [{key1: value1}, {key2:value2}]})	>db.project.find ({$and:[{Dnum:5}, {Pname:"ProductZ"}]}).pretty ();
OR	>db.collection_name.find({$or: [{key1: value1}, {key2:value2}]})	>db.project.find ({$or:[{Dnum:5}, {Pname:"ProductZ"}]}).pretty ();

7) Documents can either be embedded in one to one relation or one to many relation
   - In one to one, the new field contains only a single embedded document.
   - In one to many, the new field contains an array of multiple embedded documents.

Output: Screenshots



```
mongosh mongodb://127.0.0.
Please enter a MongoDB connection string (Default: mongodb://localhost/):

Current Mongosh Log ID: 644e77b0d8687cb7025af374
Connecting to:          mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000&appName=mongosh+1.8.0
Using MongoDB:          6.0.5
Using Mongosh:          1.8.0

For mongosh info see: https://docs.mongodb.com/mongodb-shell/

------
   The server generated these startup warnings when booting
   2023-04-30T12:55:43.624+05:30: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
------

------
   Enable MongoDB's free cloud-based monitoring service, which will then receive and display
   metrics about your deployment (disk utilization, CPU, operation statistics, etc).

   The monitoring data will be available on a MongoDB website with a unique URL accessible to you
   and anyone you share the URL with. MongoDB may use this information to make product
   improvements and to suggest MongoDB products and deployment options to you.

   To enable free monitoring, run the following command: db.enableFreeMonitoring()
   To permanently disable this reminder, run the following command: db.disableFreeMonitoring()
------

test> // Dropping the 'Company' database

test> use("Company");
switched to db Company
Company> db.dropDatabase();
{ ok: 1, dropped: 'Company' }
Company>

Company> // create a database named 'Company'

Company> use("Company");
already on db Company
Company> db.dropDatabase();
{ ok: 1, dropped: 'Company' }
```



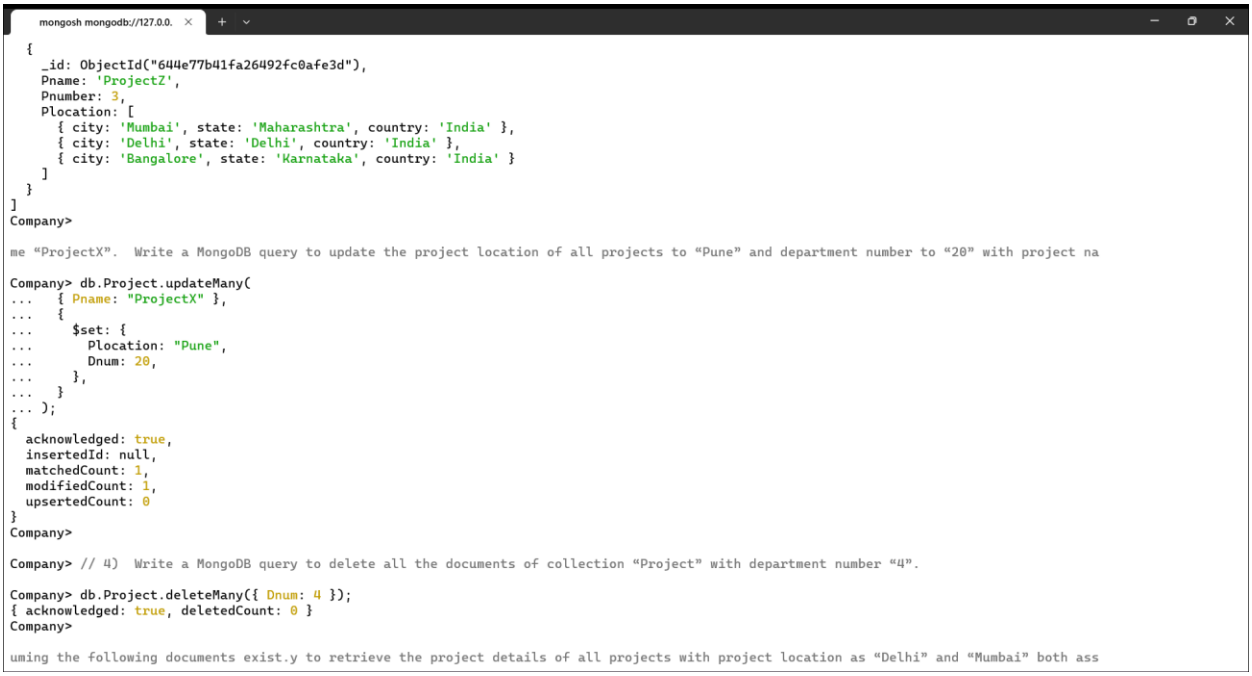
```
mongosh mongodb://127.0.0.

test> // Dropping the 'Company' database

test> use("Company");
switched to db Company
Company> db.dropDatabase();
{ ok: 1, dropped: 'Company' }
Company>

Company> // create a database named 'Company'

Company> use("Company");
already on db Company
Company> db.dropDatabase();
{ ok: 1, dropped: 'Company' }
Company>

Company> // create a collection 'Project

Company> db.createCollection("Project");
{ ok: 1 }
Company>

Company> // create a collection 'Department

Company> db.createCollection("Department");
{ ok: 1 }
Company>

Company> // insert data into the Project collection using insertOne()

Company> db.Project.insertOne({
...     Pname: "ProjectY",
...     Pnumber: 2,
...     Plocation: [
...       {
...         city: "Mumbai",
...         state: "Maharashtra",
...         country: "India",
...       },
```

```
mongosh mongodb://127.0.0.
Company> // insert data into the Project collection using insertOne()

Company> db.Project.insertOne({
...   Pname: "ProjectY",
...   Pnumber: 2,
...   Plocation: [
...     {
...       city: "Mumbai",
...       state: "Maharashtra",
...       country: "India",
...     },
...     {
...       city: "Delhi",
...       state: "Delhi",
...       country: "India",
...     },
...     {
...       city: "Bangalore",
...       state: "Karnataka",
...       country: "India",
...     },
...   ],
... });
{
  acknowledged: true,
  insertedId: ObjectId("644e77b31fa26492fc0afe3b")
}
Company>

Company> // insert data into the Project collection using insertMany()

Company> db.Project.insertMany([
...   {
...     Pname: "ProjectX",
...     Pnumber: 1,
...     Plocation: [
...       {
...         city: "Mumbai",
...         state: "Maharashtra",
...         country: "India",
```

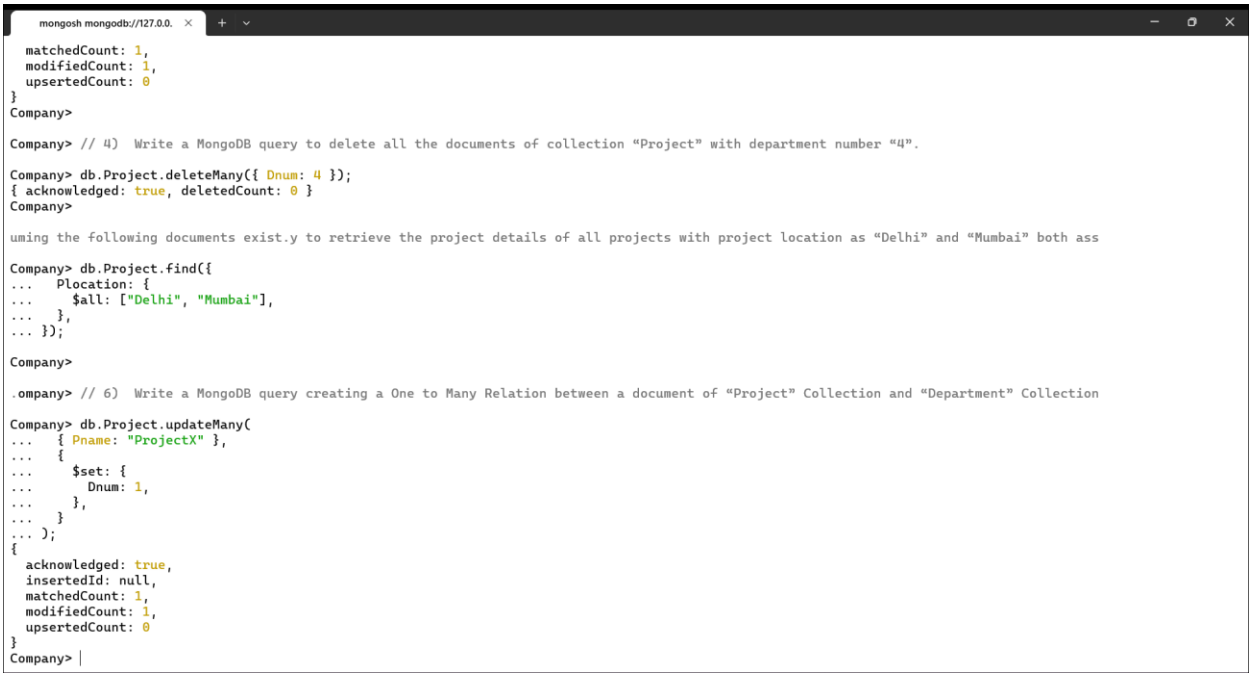
```
mongosh mongodb://127.0.0.

Company> // insert data into the Project collection using insertMany()

Company> db.Project.insertMany([
...   {
...     Pname: "ProjectX",
...     Pnumber: 1,
...     Plocation: [
...       {
...         city: "Mumbai",
...         state: "Maharashtra",
...         country: "India",
...       },
...       {
...         city: "Delhi",
...         state: "Delhi",
...         country: "India",
...       },
...       {
...         city: "Bangalore",
...         state: "Karnataka",
...         country: "India",
...       },
...     ],
...   },
...   {
...     Pname: "ProjectZ",
...     Pnumber: 3,
...     Plocation: [
...       {
...         city: "Mumbai",
...         state: "Maharashtra",
...         country: "India",
...       },
...       {
...         city: "Delhi",
...         state: "Delhi",
...         country: "India",
...       },
...       {
```

```
mongosh mongodb://127.0.0.
...         {
...           city: "Bangalore",
...           state: "Karnataka",
...           country: "India",
...         },
...       ],
...     },
... ]);
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId("644e77b41fa26492fc0afe3c"),
    '1': ObjectId("644e77b41fa26492fc0afe3d")
  }
}
Company>

Company> // insert data into the Department collection using insertOne()

Company> db.Department.insertOne({
...     Dno: 1,
...     Dname: "DepartmentX",
...     Dmanager: "ABC",
... });
{
  acknowledged: true,
  insertedId: ObjectId("644e77b41fa26492fc0afe3e")
}
Company>

Company> // insert data into the Department collection using insertMany()

Company> db.Department.insertMany([
...   {
...     Dno: 2,
...     Dname: "DepartmentY",
...     Dmanager: "DEF",
...   },
...   {
...     Dno: 3,
```

```
mongosh mongodb://127.0.0.
...   {
...     Dno: 3,
...     Dname: "DepartmentZ",
...     Dmanager: "GHI",
...   },
... ]);
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId("644e77b41fa26492fc0afe3f"),
    '1': ObjectId("644e77b41fa26492fc0afe40")
  }
}
Company>

Company> // 2)  Write the “Select * from Project” equivalent query in MongoDB.

Company> db.Project.find({});
[
  {
    _id: ObjectId("644e77b31fa26492fc0afe3b"),
    Pname: 'ProjectY',
    Pnumber: 2,
    Plocation: [
      { city: 'Mumbai', state: 'Maharashtra', country: 'India' },
      { city: 'Delhi', state: 'Delhi', country: 'India' },
      { city: 'Bangalore', state: 'Karnataka', country: 'India' }
    ]
  },
  {
    _id: ObjectId("644e77b41fa26492fc0afe3c"),
    Pname: 'ProjectX',
    Pnumber: 1,
    Plocation: [
      { city: 'Mumbai', state: 'Maharashtra', country: 'India' },
      { city: 'Delhi', state: 'Delhi', country: 'India' },
      { city: 'Bangalore', state: 'Karnataka', country: 'India' }
    ]
  },
  {
```



```
mongosh mongodb://127.0.0.
  {
    _id: ObjectId("644e77b41fa26492fc0afe3d"),
    Pname: 'ProjectZ',
    Pnumber: 3,
    Plocation: [
      { city: 'Mumbai', state: 'Maharashtra', country: 'India' },
      { city: 'Delhi', state: 'Delhi', country: 'India' },
      { city: 'Bangalore', state: 'Karnataka', country: 'India' }
    ]
  }
]
Company>

me “ProjectX”.  Write a MongoDB query to update the project location of all projects to “Pune” and department number to “20” with project na

Company> db.Project.updateMany(
...   { Pname: "ProjectX" },
...   {
...     $set: {
...       Plocation: "Pune",
...       Dnum: 20,
...     },
...   }
... );
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
Company>

Company> // 4)  Write a MongoDB query to delete all the documents of collection “Project” with department number “4”.

Company> db.Project.deleteMany({ Dnum: 4 });
{ acknowledged: true, deletedCount: 0 }
Company>

uming the following documents exist.y to retrieve the project details of all projects with project location as “Delhi” and “Mumbai” both ass
```



```
mongosh mongodb://127.0.0.
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
Company>

Company> // 4)  Write a MongoDB query to delete all the documents of collection “Project” with department number “4”.

Company> db.Project.deleteMany({ Dnum: 4 });
{ acknowledged: true, deletedCount: 0 }
Company>

uming the following documents exist.y to retrieve the project details of all projects with project location as “Delhi” and “Mumbai” both ass

Company> db.Project.find({
...   Plocation: {
...     $all: ["Delhi", "Mumbai"],
...   },
... });

Company>

.ompany> // 6)  Write a MongoDB query creating a One to Many Relation between a document of “Project” Collection and “Department” Collection

Company> db.Project.updateMany(
...   { Pname: "ProjectX" },
...   {
...     $set: {
...       Dnum: 1,
...     },
...   }
... );
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
Company>
```

Preparatory Questions

Q1) The order of documents returned by a query is not defined unless you specify a _____

1) sortfind()
2) sortelse()
3) sort()
4) none of the mentioned

Q2) In aggregation pipeline, the _____ pipeline stage provides access to MongoDB queries.

1) $catch
2) $match
3) $batch
4) All of the mentioned

Q3) Point out the wrong statement.

1) sort() modifier sorts the results by age in ascending order
2) Queries in MongoDB return all fields in all matching documents by default
3) To scale the amount of data that MongoDB sends to applications, include a projection in the queries.
4) None of the mentioned

Q4) Which of the following query selects documents in the records collection that match the condition { “user_id”: { $lt: 42 } }?

1) db.records.findOne({ “user_id”: { $lt: 42 } }, { “history”: 0 })
2) db.records.find({ “user_id”: { $lt: 42 } }, { “history”: 0 })
3) db.records.findOne({ “user_id”: { $lt: 42 } }, { “history”: 1 })
4) db.records.select({ “user_id”: { $lt: 42 } }, { “history”: 0 })

Q5) Which of the following is not a projection operator?

1) $slice
2) $elemMatch
3) $
4) None of the mentioned

Value Added Experiment

Student Name and Roll Number:
Semester /Section:
Link to Code:
Date:
Faculty Signature:
Marks:

Objective

Design an ER/ EER Diagram, relational schema and implement the database in MongoDB.

Program Outcome

- The students will design and implement DBMS concepts on a real-world project.

Problem Statement

The NorthCap University has recently launched new specializations in B-Tech (Computer Science & Engineering). To facilitate this, a new admission application needs to be developed, where an Admin should be able to create a new application, update an existing application and delete any application. The application should contain the below mentioned fields:

1) Student Name
2) Student Id
3) Father's Name
4) Mother's Name
5) Gender
6) DOB
7) Nationality
8) Address
9) Marks in Class X. XII and JEE.
10) Email
11) Contact No.
12) B-Tech CSE Specialization Choice 1 (IOT, Data Science, Cyber Security, Full Stack and Game Tech)

13) B-Tech CSE Specialization Choice 2 (IOT, Data Science, Cyber Security, Full Stack and Game Tech)

Based on the application requirements and your specific assumptions, draw an ER/EER model for the application's database and further map it to appropriate relational schema.

Database for B.Tech Specialization

Collections

1. Student

Possible Fields:

- Student Name (Sname)
- Student Id should be unique (Sid)
- Address
  - Address can be an array of multiple address(or)
  - Address can be a composite field with sub field-city, state, country
- Marks in JEE. (Marks_JEE)
- B-Tech CSE Specialization Choice 1 (Choise_1)
- B-Tech CSE Specialization Choice 2 (Choise_2)

2. Specialization

Possible Fields:

- B-Tech CSE Specialization Choice IOT (IOT)
- B-Tech CSE Specialization Data Science (Data_Science)
- B-Tech CSE Specialization Cyber Security (Cyber_Security)
- B-Tech CSE Specialization Full Stack (Full_Stack)
- B-Tech CSE Specialization Game Tech (Game_Tech)

Queries:

1. Insert multiple documents (2-3) together in the Student collection with at least 1 student Address as a composite attribute.
2. Write the "Select * from Student" equivalent query in MongoDB.

3. Write a MongoDB query to return the formatted/structured Student details of all Students with Specialization choice 1 “IOT” and Student name as “RAM”.

4. Write a MongoDB query to return the formatted/structured Student details (including Student name, choice1 and choice2 and excluding other fields) of all students with choice1 as “IOT” or Student name as “RAM”.

5. Write a MongoDB query to return the formatted/structured Student details of all students with choice1 equal to “Full_Stack” and Student JEE Marks less than 200 or student address as “Delhi”.

6. Write a MongoDB query to update the student choice1 to “IOT” and JEE Marks to “80” with student name “XYZ”.

7. Write a MongoDB query to delete all the documents of collection “student” with choice1 as “IOT”.

8. Write a MongoDB query to retrieve the Student details of all students with address as “Delhi” and “Mumbai”.

9. Write a MongoDB query creating a One to Many Relation between a document of“student” Collection and “Specialization” Collection.

10. Write a MongoDB query to retrieve the student details student name, Marks_JEE, of student with specialization as Cyber_Security (for department in which the project is being run) as “ABC”.

Output: Screenshots