3/13/2023

Name: Aakash Giri  
SID: 13898274

Name: Sunil Sapkota  
SID: 13899053

Database Management System

Softwarica College of IT and e-commerce

Contents

[Part A: ER Diagram 2](#_Toc130108415)

[Task 1 and 2 2](#_Toc130108416)

[Part B: SQL Programming 4](#_Toc130108417)

[Task 1: 4](#_Toc130108418)

[Task 2: a 6](#_Toc130108419)

[Task 2: b 7](#_Toc130108420)

[Task 2: c 8](#_Toc130108421)

[Task 2: d 9](#_Toc130108422)

[Task C: Sequential and Distributed Processing 10](#_Toc130108423)

[Task C: 1 10](#_Toc130108424)

[Task C: 2 12](#_Toc130108425)

[The advantage and benefit of using MapReduce are given below: 13](#_Toc130108426)

[Task D: 14](#_Toc130108427)

[Advantages of relational databases (MySQL): 14](#_Toc130108428)

[Disadvantages of relational databases (MySQL): 14](#_Toc130108429)

[Advantages of NoSQL databases (MongoDB): 15](#_Toc130108430)

[Disadvantages of NoSQL databases (MongoDB): 15](#_Toc130108431)

[Conclusion: 16](#_Toc130108432)

[Reference: 16](#_Toc130108433)

# Part A: ER Diagram

Diagram

Description automatically generated

Figure 1: E R diagram of task A

## Task 1 and 2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Employee | | | | |
| \*emp\_id | first\_name | last\_name | phone\_nu | past\_history |

Given scenario at task 1 can be represented by constructing six table, employee, department, jobtype, review, worker and supervise.

Table employee consist emp\_id attribute as varchar (string), primary key represent by \* symbol in table which store the unique id of employee. The first\_name and last\_name store the first name and last name of the employee respectively are varchar length 40 each. Credential phone number is store in phone\_nu attribute which is multivalued attribute and is integer with and the past history is store in attribute past\_history with varchar.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| worker | | | | |
| \*work\_id | emp\_id | work\_type | talent | duty |

The worker table contain work\_id as primary attribute and varchar. It is linked with one to one relationship with employee table with foreign key as emp\_id which is a varchar. It contains work\_type as attribute it store the data to employee whether he work in floor or office. Attribute like talent and duty are all string variable to store the talent and duty of the worker.

|  |  |  |
| --- | --- | --- |
| Department | | |
| \*dep\_id | emp\_id | dep\_name |

Table department has three attribute, dep\_id is a varchat with length 10 and a primary key. Department name is store in attribute dep\_name as varchar of length 40. It is linked with employee table with the foreign key emp\_id. The relation between both tables is a one-to-many relationship which show that one department contain many employee.

|  |  |  |  |
| --- | --- | --- | --- |
| job | | | |
| \*job\_id | #emp\_id | position | grade |

Job position and grade are store in another table name job in attribute job\_name and grade respectively both are varchar of length 40. It consists of emp\_id as foreign key to link the job table to the employee table. One to one relationship exists between both table.

|  |  |  |  |
| --- | --- | --- | --- |
| review | | | |
| \*review\_id | #emp\_id | #reviewed\_by | date |

The data of review of the employee by reviewer is store in the attribute in emp\_id and reviewed\_by respectively and attributes are varchar of length 10, both these are foreign key to employee and department table. It has one to many relationship with employee and department table which show that the many record of one employee and department can be found in review table. It contains the date of the review in attribute date.

|  |  |  |  |
| --- | --- | --- | --- |
| supervise | | | |
| \*super\_id | #mag\_id | #ass\_mag | employee |

The supervise table contain four attribute super\_id as primary key and it’s store the string datatype as varchar. It contain emp\_id as a linked between department table and has a relation of one to one relationship. It contain mag-id and ass\_mag to store the value of the manager id and assistant manager id.

# Part B: SQL Programming

## Task 1:

-- Creating table Employee

CREATE TABLE Employee (

empid VARCHAR(10) PRIMARY KEY,

name VARCHAR(50),

address VARCHAR(50),

DOB DATE,

job VARCHAR(50),

salaryCode VARCHAR(10),

depid VARCHAR(10),

manager VARCHAR(50) ,

schemeid VARCHAR(10)

);

-- Creating table Department

CREATE TABLE Department (

depid VARCHAR(10) PRIMARY KEY,

name VARCHAR(50)

);

-- Creating table SalaryGrade

CREATE TABLE SalaryGrade (

salaryCode VARCHAR(10) PRIMARY KEY,

startSalary INT(10),

finishSalary INT(10)

);

-- Creating table PensionScheme

CREATE TABLE PensionScheme (

schemeid VARCHAR(10) PRIMARY KEY,

name VARCHAR(50),

rate DECIMAL(5,2)

);

-- Altering the foreign key of employee table

ALTER TABLE Employee ADD FOREIGN KEY(depid) REFERENCES Department(depid) ON DELETE SET NULL;

ALTER TABLE Employee ADD FOREIGN KEY(schemeid) REFERENCES PensionScheme(schemeid) ON DELETE SET NULL;

ALTER TABLE Employee ADD FOREIGN KEY(salaryCOde) REFERENCES SalaryGrade(salaryCode) ON DELETE SET NULL;

-- First inserting the foregien key table data

INSERT INTO Department VALUES

('D10', 'Administration'),

('D20', 'Finance'),

('D30', 'Sales'),

('D40', 'Maintenance'),

('D50', 'IT Support');

INSERT INTO PensionScheme VALUES

('S110', 'AXA' ,0.5),

( 'S121', 'Premier', 0.6),

( 'S124' ,'Stakeholder', 0.4),

('S116', 'Standard' ,0.4);

INSERT INTO SalaryGrade VALUES

('S1' ,15000 ,18000),

('S2' ,18001 ,22000),

('S3', 22001 ,25000),

('S4', 25001, 29000),

('S5', 29001, 38000);

-- Inserting the data in table employee

INSERT INTO employee VALUES

('E101', 'Young, S.', '199 London Road', '2076/03/05' ,'Clerk' ,'S1', 'D10', 'E110', 'S116'),

('E301', 'April, H.', '20 Glade close', '2079/03/10' ,'Sales Person', 'S2', 'D30', 'E310', 'S124'),

('E310', 'Newgate, E.', '10 Heap Street', '2080/11/28' ,'Manager', 'S5', 'D30', NULL ,'S121'),

('E501', 'Teach, E', '22 railway road', '2072/02/12', 'Analyst' ,'S5', 'D50', NULL ,'S121'),

('E102', 'Hawkins, M.', '3 High Street', '2074/07/13', 'Clerk', 'S1', 'D10', 'E110', 'S116'),

('E110', 'Watkins, J.',' 11 crescent road' ,'2069/06/25', 'Manager', 'S5', 'D10', NULL ,'S121');

## Task 2: a

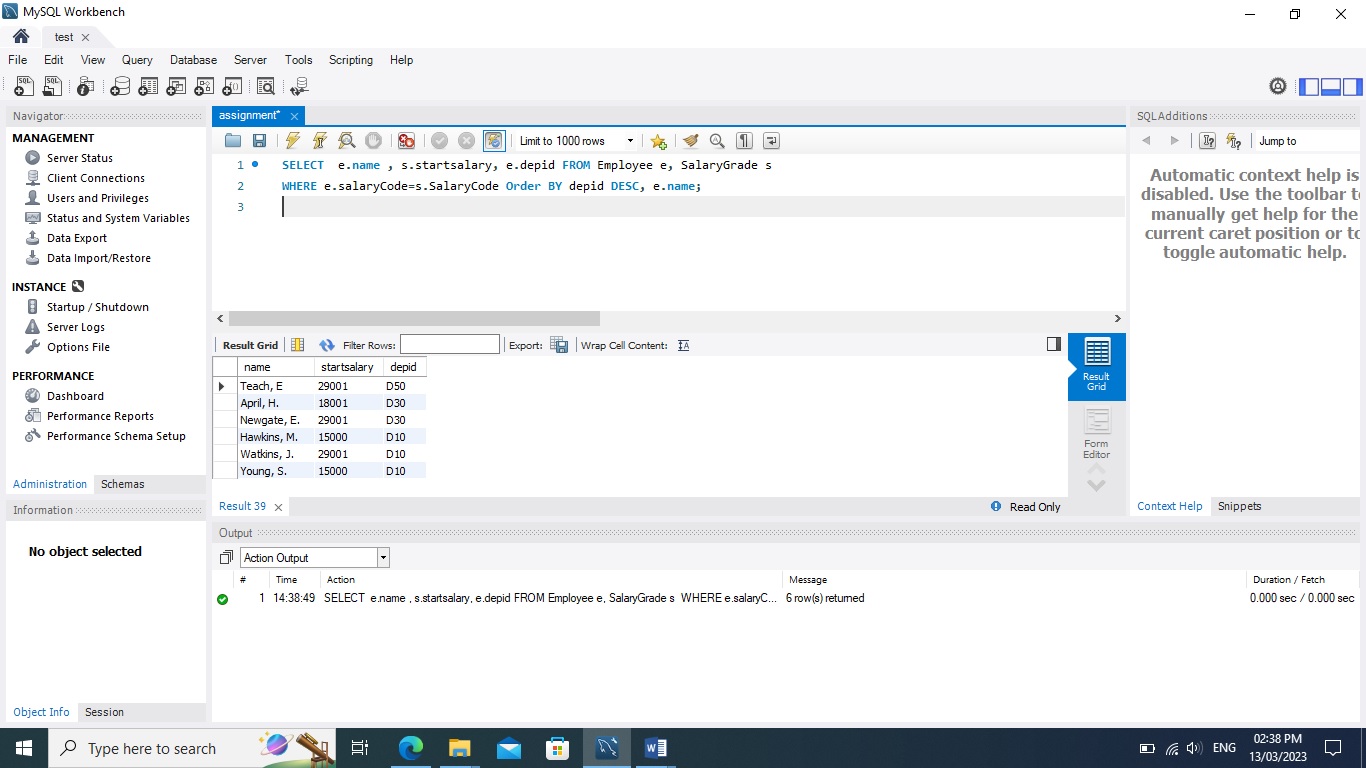
1. The name (in ascending order), the starting salary and department id of each employee within a descending order of department ids.

For this query we use select name, startsalary and depid from table employee and salarygrade putting condition using where salary code of employee = salarycode of salarygrade and order be depid in descending order and ascending order by employee name.

SELECT e.name , s.startsalary, e.depid FROM Employee e, SalaryGrade s

WHERE e.salaryCode=s.SalaryCode Order BY depid DESC, e.name;

|  |  |  |
| --- | --- | --- |
| **name** | **startsalary** | **depid** |
| Teach, E | 29001 | D50 |
| April, H. | 18001 | D30 |
| Newgate, E. | 29001 | D30 |
| Hawkins, M. | 15000 | D10 |
| Watkins, J. | 29001 | D10 |
| Young, S. | 15000 | D10 |



## Task 2: b

1. Give the number of employees for each of the pension schemes offered by the company. Result listing should include the name of each scheme and its corresponding number of employees who join the scheme.

In this second query we use select name from pensionscheme and use count function to schemeid of employee using where condition to check schemeid of pensionscheme and employee is equal and the grouping the result on schemeid of employee.

SELECT p.name, COUNT(e.schemeid) numberofemp From Pensionscheme p, Employee e WHERE p.schemeid =e.schemeid GROUP BY e.schemeid;

|  |  |
| --- | --- |
| **name** | **numberofemp** |
| Standard | 2 |
| Premier | 3 |
| Stakeholder | 1 |

Graphical user interface, text, application, email

Description automatically generated

## Task 2: c

1. Give the total number of employees who are not managers but currently receive an annual salary of over £35,000.

First we select name and empid from employee and checking condition using where and forming a sub query selecting salarycode from table salarygrade where finishsalary>35000. The given result is the compare with salarycode of employee. The salarycode of both should be equal. And we cancel the result which match empid E110 and E310 from employee table.

SELECT name, e.empid FROM Employee e where (SELECT s.SalaryCode FROM SalaryGrade s WHERE FinishSalary >35000 ) =e.SalaryCode AND e.empid!= ('E110') And e.empid!= ('E310') ;

|  |  |
| --- | --- |
| name | empid |
| Teach, E | E501 |

Graphical user interface, text, application, email

Description automatically generated

## Task 2: d

1. List the id and name of each employee along with his/her manager’s name.

For this query we select empid and name from employee table and name from table tt as manager\_ name. Table tt is subquery where we select empid and name from employee where managerid is null and join both table on manager from employee = to empid from table tt.

SELECT e.empid, e.name, tt.name AS Manager\_name FROM Employee e

JOIN

(SELECT a.empid, a.name FROM Employee a WHERE a.manager is null) AS tt ON e.manager = tt.empid ;

|  |  |  |
| --- | --- | --- |
| **empid** | **name** | **Manager\_name** |
| E101 | Young, S. | Watkins, J. |
| E102 | Hawkins, M. | Watkins, J. |
| E301 | April, H. | Newgate, E. |

Graphical user interface, text, application, email

Description automatically generated

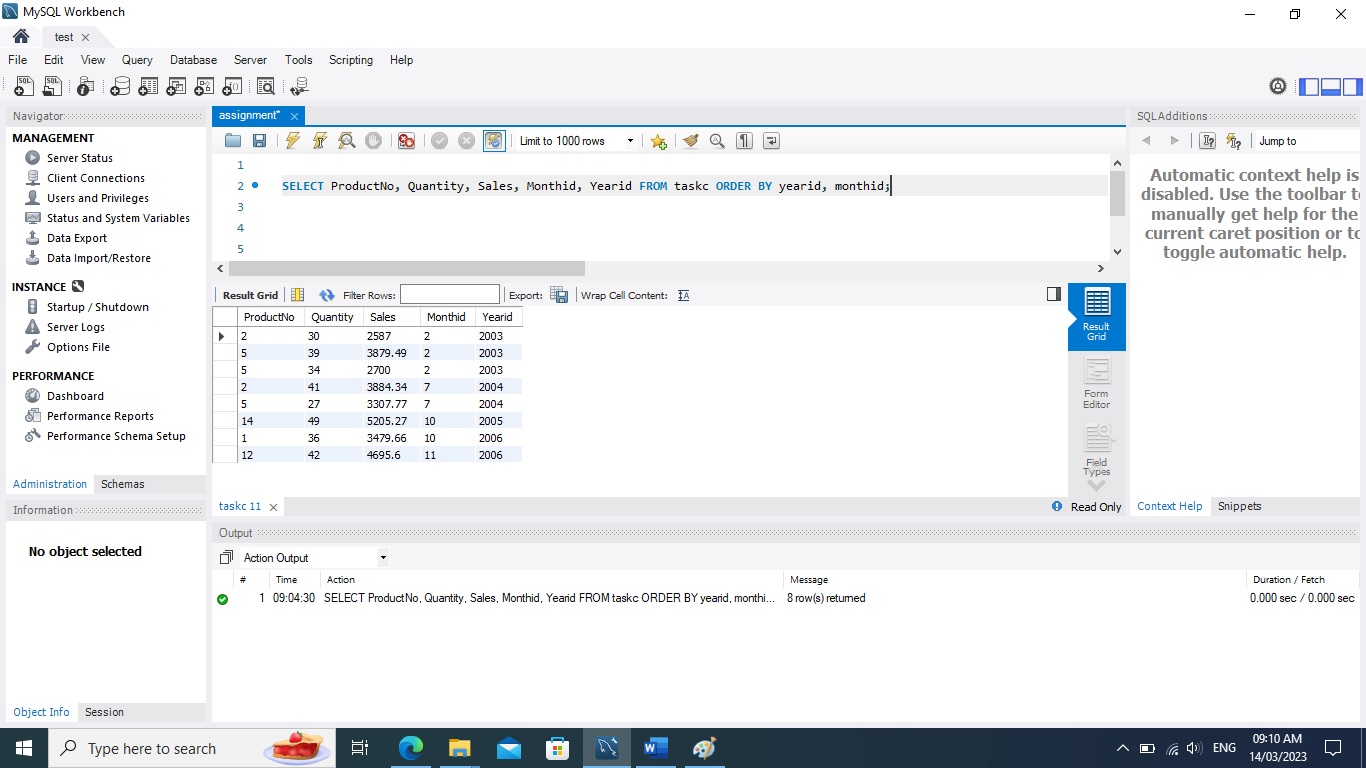
# Task C: Sequential and Distributed Processing

## Task C: 1

Here we select productno, quality, sales, monthid and yearid from table name taskc and order the given result by first monthid and yearid in ascending order.

SELECT ProductNo, Quantity, Sales, Monthid, Yearid FROM taskc ORDER BY yearid, monthid;

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ProductNo** | **Quantity** | **Sales** | **Monthid** | **Yearid** |
| 2 | 30 | 2587 | 2 | 2003 |
| 5 | 39 | 3879.49 | 2 | 2003 |
| 5 | 34 | 2700 | 2 | 2003 |
| 2 | 41 | 3884.34 | 7 | 2004 |
| 5 | 27 | 3307.77 | 7 | 2004 |
| 14 | 49 | 5205.27 | 10 | 2005 |
| 1 | 36 | 3479.66 | 10 | 2006 |
| 12 | 42 | 4695.6 | 11 | 2006 |



## Task C: 2

When dealing with large datasets that are too big for centralised processing in a relational database, MapReduce help to break down the data and process it in a decentralized manner. MapReduce is a programming model that allows the parallel processing of large datasets across distributed clusters which consist of two main phases they are the map phase where the input data is divided into smaller chunks and processed independently and the reduce phase where the results from the map phase are aggregated to produce the final output.

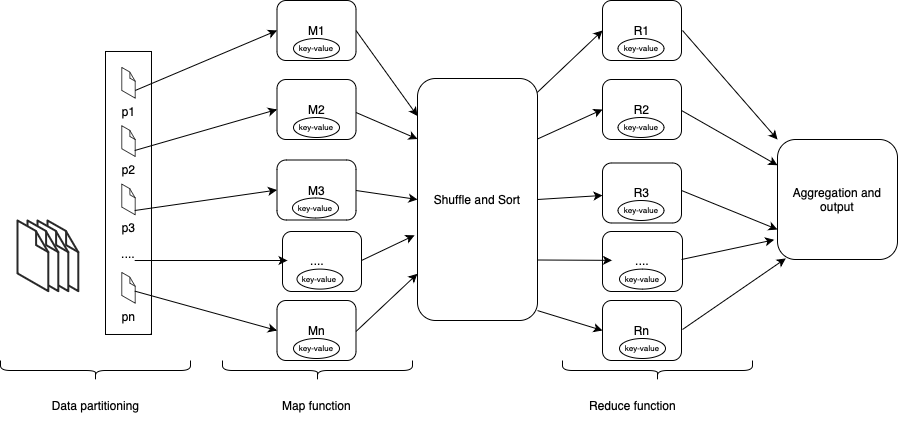
Following steps are the part of MapReduce:

* **Data partitioning:** Divide the input data file into smaller chunks or partitions, ensuring that each chunk is small enough to be processed by a single node in the cluster. The partitioning strategy should aim to minimize data skew and ensure balanced workload distribution across the nodes.
* **Map function:** Design a map function that takes a single partition as input, processes the data and emits key-value pairs as intermediate results. The specific details of the map function depend on the problem you are trying to solve. The key-value pairs should be chosen in a way that helps the reduce function to easily aggregate the results.
* **Shuffle and Sort:** After the map phase the intermediate key-value pairs need to be shuffled and sorted so that all the values for a given key are grouped together. This is an essential step as it prepares the data for the reduce phase.
* **Reduce function:** Design a reduce function that takes a key and its associated list of values as input and process them and emits a final result.
* **Aggregation and output:** Collect the results from all the reduce nodes and combine them to generate the final output. This may involve additional processing, depending on the problem you are trying to solve.

### The advantage and benefit of using MapReduce are given below:

* **Scalability:** MapReduce is designed to handle large datasets by leveraging the power of distributed processing. It can scale horizontally by adding more nodes to the cluster, allowing it to process even larger datasets.
* **Fault tolerance:** MapReduce automatically handles node failures, ensuring that the processing continues without interruption. If a node fails during the Map or Reduce phase, the work is automatically reassigned to another node.
* **Parallelism:** MapReduce allows multiple tasks to run simultaneously across multiple nodes, reducing the overall processing time.
* **Flexibility:** MapReduce can be used to process both structured and unstructured data, making it suitable for a wide range of applications.

In summary, a decentralized MapReduce solution is an effective approach for processing large datasets that cannot be handled by a centralized relational database. It offers scalability, fault tolerance, and parallelism, while being flexible enough to handle various data types and problem domains.



## Task D:

Craigslist to decide to transition from relational MySQL servers to NoSQL MongoDB servers in 2011. We can comprehend the rationale behind this move by contrasting relational and NoSQL databases' benefits and drawbacks.

### Advantages of relational databases (MySQL):

* **Structured data:** Relational databases use a fixed schema which enforces a well-defined structure for the data. This ensures consistency and makes it easier to establish relationships between different data elements.
* **ACID properties:** Relational databases adhere to the ACID properties (Atomicity, Consistency, Isolation, Durability), ensuring that transactions are processed reliably.
* **Standardized query language:** SQL (Structured Query Language) is a standardized language used for querying relational databases, making it easier to work with data across different platforms.
* **Prevents Data Redundancy:** With the feather like primary key, constrain like unique and feather like normalization it prevents the data duplication and improve data integrity, consistency and efficiency.

### Disadvantages of relational databases (MySQL):

* **Scalability:** Relational databases may struggle to scale horizontally, as they are generally optimized for single-server deployment. This can lead to performance issues when dealing with very large datasets or high throughput rates and require expensive hardware upgrades.
* **Schema rigidity:** Altering the schema of a relational database can be time-consuming and expensive, especially when dealing with large amounts of data. This inflexibility makes it difficult to adapt to evolving data requirements or to introduce new features. It cannot store unstructured data or semi-structured data such as multimedia files or social media files or metadata.
* **Complexity:** Relational databases can become complex and difficult to manage as the number of tables and relationships between them increases and it required significant time, effort and resources to maintaining an complex database.

### Advantages of NoSQL databases (MongoDB):

* **Scalability:** NoSQL databases like MongoDB are designed for horizontal scalability, allowing them to distribute data across multiple servers and handle large amounts of data and traffic more efficiently.
* **Schema flexibility:** MongoDB uses a dynamic schema, which allows for the storage of diverse data structures without the need for predefined schemas. This makes it easier to modify the data model and add new features to the system.
* **High availability:** MongoDB provides built-in replication and automatic failover, ensuring high availability and fault tolerance in the event of server failures.

### Disadvantages of NoSQL databases (MongoDB):

* **Lack of ACID properties:** NoSQL databases may not fully support ACID properties, which can lead to potential inconsistencies in the data. However, MongoDB does offer tuneable consistency levels, allowing users to balance consistency with performance according to their needs.
* **Less mature ecosystem:** NoSQL databases are generally newer than relational databases, which means that their ecosystems may not be as mature, and finding experienced developers or support may be more challenging.

Comparing the pros and cons of rdbms and nosql we can explain why craigslist switched from rdbms (MYSQL) to nosql (MongoDB).

Handling large volumes of data: With over 1.5 million new advertisements posted daily, Craigslist needed a solution that could efficiently handle the massive amount of data generated. MongoDB's horizontal scalability allowed for better performance in managing such large datasets.

Evolving data structure: Craigslist's data structure changed multiple times over the years. The rigid schema of MySQL made it difficult and time-consuming to alter the database structure. MongoDB's flexible schema allowed Craigslist to easily adapt to changing data requirements without causing disruptions or outages.

Ease of adding new features: With MongoDB's schema flexibility, it became easier for Craigslist to add new functionality to the system without the need for extensive modifications to the database.

Archiving challenges: The statutory requirement to transfer documents to an archival area after 60 days added complexity to the system. MongoDB's flexible schema simplified the process of synchronizing changes between the live database and the archive, reducing the time taken for such operations.

In conclusion, Craigslist's switch from MySQL to MongoDB was driven by the need for a more scalable and flexible database solution to handle the large volumes of data and the evolving requirements of the system. MongoDB's horizontal scalability, dynamic schema, and ease of adding new features made it an ideal.

# Conclusion:

In conclusion this assignment gives us a opportunity to learn SQL, Hadoop eco system and MangoDB. We perform our task in MYSQL which is a free rdbms and support all SQL language. MangoDB is NOSQL language which share the code of JavaScript and its gaining huge popularity due to horizontal scalability and flexibility.

# Reference:

Simplilearn.com. (n.d.). *ER Diagrams in DBMS: Entity Relationship Diagram Model*. [online] Available at: <https://www.simplilearn.com/tutorials/sql-tutorial/er-diagram-in-dbms>.

www.javatpoint.com. (n.d.). *Learn MongoDB Tutorial - javatpoint*. [online] Available at: <https://www.javatpoint.com/mongodb-tutorial>.

w3schools (2019). *SQL Tutorial*. [online] W3schools.com. Available at: <https://www.w3schools.com/sql/>.

www.javatpoint.com. (n.d.). *Learn Big Data Hadoop Tutorial - javatpoint*. [online] Available at: <https://www.javatpoint.com/hadoop-tutorial>.

Spiceworks. (n.d.). *What Is MapReduce? Meaning, Working, Features, and Uses*. [online] Available at: <https://www.spiceworks.com/tech/big-data/articles/what-is-map-reduce/>.

Anderson, B. and Nicholson, B. (2021). *SQL vs. NoSQL Databases: What’s the Difference?* [online] www.ibm.com. Available at: <https://www.ibm.com/cloud/blog/sql-vs-nosql>.

‌

‌

‌

‌

‌