**Task1**

**1.**

**2.** "Database Management Systems" by Raghu Ramakrishnan and Johannes Gehrke

"Database Systems: The Complete Book" by Hector Garcia-Molina, Jeffrey D. Ullman, and Jennifer Widom

"Fundamentals of Database Systems" by Ramez Elmasri and Shamkant Navathe

**3.** However, based on current trends and advancements, it is expected that database systems in 2023 will continue to evolve towards cloud-based and hybrid systems, with increased use of NoSQL and NewSQL databases for big data processing and real-time analytics, while also incorporating more advanced security and privacy measures to protect sensitive data.

**Task 2**

1. <https://www.guru99.com/best-database-management-software.html>

1) ManageEngine Applications Manager – Best overall database tracking software.

2) DbVisualizer – Best for SQL database analysis and development.

3) Plan Explorer – SQL Query Analysis.

4) MySQL – Best For Managing business-critical SQL applications.

5) Oracle Database – Best for its security and reliability.

6) Valentina Studio – Best For Small organizations with limited budgets

7) Microsoft SQL Server – Best For comprehensive database management solution

8) IBM DB2 – Best for running mission-critical workloads for businesses

9) Hadoop HDFS – Best for Parallel processing

10) PHPMyAdmin – Best for Administration of MySQL over the internet

11) MongoDb – Best for Development and scaling

**Task 4**

1. Find all employees who have a salary greater than the average salary of their department:

SELECT \* FROM employees

WHERE salary > (

SELECT AVG(salary) FROM employees

WHERE department\_id = employees.department\_id

);

1. Find all departments that have at least one employee with a salary greater than $100,000:

SELECT \* FROM departments

WHERE department\_id IN (

SELECT department\_id FROM employees

WHERE salary > 100000

);

1. Find the highest salary in each department:

SELECT department\_id, MAX(salary) FROM employees

GROUP BY department\_id;

1. Find all employees who have a manager with a salary greater than $150,000:

SELECT \* FROM employees

WHERE manager\_id IN (

SELECT employee\_id FROM employees

WHERE salary > 150000

);

1. Find all employees who have the same last name as their manager:

SELECT \* FROM employees

WHERE last\_name = (

SELECT last\_name FROM employees

WHERE employee\_id = employees.manager\_id

);

1. Find all employees who have no direct reports:

SELECT \* FROM employees

WHERE employee\_id NOT IN (

SELECT manager\_id FROM employees

);

1. Find all employees who have more than one direct report:

SELECT \* FROM employees

WHERE employee\_id IN (

SELECT manager\_id FROM employees

GROUP BY manager\_id

HAVING COUNT(\*) > 1

);

1. Find all employees who have been with the company for less than 5 years:

SELECT \* FROM employees

WHERE hire\_date > (

SELECT DATEADD(year, -5, GETDATE())

);

1. Find all employees who have a bonus greater than 10% of their salary:

SELECT \* FROM employees

WHERE salary \* 0.1 < (

SELECT bonus FROM bonuses

WHERE employee\_id = employees.employee\_id

);

1. Find all employees who have the same job title as their manager:

SELECT \* FROM employees

WHERE job\_title = (

SELECT job\_title FROM employees

WHERE employee\_id = employees.manager\_id

);

**Task 5 a**

1Assuming you are using a relational database management system (RDBMS) such as MySQL or PostgreSQL, here are the steps to create a database schema:

1. Open your RDBMS and create a new database.

CREATE DATABASE mydatabase;

1. Create tables for your database, defining the columns and data types for each table.

CREATE TABLE customers (

id INT PRIMARY KEY,

first\_name VARCHAR(50),

last\_name VARCHAR(50),

email VARCHAR(50)

);

CREATE TABLE orders (

id INT PRIMARY KEY,

customer\_id INT,

product\_name VARCHAR(50),

quantity INT,

price DECIMAL(10, 2),

FOREIGN KEY (customer\_id) REFERENCES customers(id)

);

1. Insert at least 15 records into each table.

INSERT INTO customers VALUES

(1, 'John', 'Doe', 'johndoe@email.com'),

(2, 'Jane', 'Smith', 'janesmith@email.com'),

...

;

INSERT INTO orders VALUES

(1, 1, 'Widget', 10, 50.00),

(2, 1, 'Gadget', 5, 25.00),

...

;

1. Perform at least 5 queries using the SELECT command.

SELECT \* FROM customers;

SELECT \* FROM orders;

SELECT \* FROM customers WHERE last\_name = 'Smith';

SELECT \* FROM orders WHERE price > 100.00;

SELECT c.first\_name, c.last\_name, SUM(o.quantity \* o.price) AS total\_spent

FROM customers c

JOIN orders o ON c.id = o.customer\_id

GROUP BY c.id;

1. Modify at least one entry in each table.

UPDATE customers SET email = 'newemail@email.com' WHERE id = 1;

UPDATE orders SET quantity = 7 WHERE id = 2;

1. Formulate at least 5 queries each with a different aggregated value.

-- Count the number of customers

SELECT COUNT(\*) FROM customers;

-- Get the average price of an order

SELECT AVG(price) FROM orders;

-- Get the total quantity of a product sold

SELECT SUM(quantity) FROM orders WHERE product\_name = 'Widget';

-- Get the minimum price of an order

SELECT MIN(price) FROM orders;

-- Get the maximum quantity of a product sold

SELECT MAX(quantity) FROM orders WHERE product\_name = 'Gadget';

1. Enforce referential integrity in at least two tables.

Referential integrity can be enforced by defining foreign keys in child tables that reference the primary keys in the parent tables. This ensures that any data entered into the child table references valid data in the parent table.

In the example schema above, referential integrity is enforced in the "orders" table with the foreign key constraint:

FOREIGN KEY (customer\_id) REFERENCES customers(id)

This ensures that any value entered into the "customer\_id" column of the "orders" table exists in the "id" column of the "customers" table.

You can enforce referential integrity in other tables by creating similar foreign key constraints that reference the appropriate primary keys in the parent tables.

**Task 5 b**

Sure, here is an example schema for a product orders database in Oracle with tables for Customer, Invoice, Line, Product, and Vendor:

CREATE TABLE CUSTOMER (

CUSTOMER\_ID INT PRIMARY KEY,

NAME VARCHAR(50) NOT NULL,

ADDRESS VARCHAR(100) NOT NULL,

CITY VARCHAR(50) NOT NULL,

STATE VARCHAR(50) NOT NULL,

ZIP VARCHAR(20) NOT NULL,

COUNTRY VARCHAR(50) NOT NULL,

PHONE VARCHAR(20) NOT NULL,

EMAIL VARCHAR(50) NOT NULL

);

CREATE TABLE VENDOR (

VENDOR\_ID INT PRIMARY KEY,

NAME VARCHAR(50) NOT NULL,

ADDRESS VARCHAR(100) NOT NULL,

CITY VARCHAR(50) NOT NULL,

STATE VARCHAR(50) NOT NULL,

ZIP VARCHAR(20) NOT NULL,

COUNTRY VARCHAR(50) NOT NULL,

PHONE VARCHAR(20) NOT NULL,

EMAIL VARCHAR(50) NOT NULL

);

CREATE TABLE PRODUCT (

PRODUCT\_ID INT PRIMARY KEY,

NAME VARCHAR(50) NOT NULL,

DESCRIPTION VARCHAR(100) NOT NULL,

CATEGORY VARCHAR(50) NOT NULL,

PRICE DECIMAL(8,2) NOT NULL,

VENDOR\_ID INT NOT NULL,

FOREIGN KEY (VENDOR\_ID) REFERENCES VENDOR(VENDOR\_ID) ON DELETE CASCADE

);

CREATE TABLE INVOICE (

INVOICE\_ID INT PRIMARY KEY,

CUSTOMER\_ID INT NOT NULL,

INVOICE\_DATE DATE NOT NULL,

TOTAL\_AMOUNT DECIMAL(8,2) NOT NULL,

FOREIGN KEY (CUSTOMER\_ID) REFERENCES CUSTOMER(CUSTOMER\_ID) ON DELETE CASCADE

);

CREATE TABLE LINE (

LINE\_ID INT PRIMARY KEY,

INVOICE\_ID INT NOT NULL,

PRODUCT\_ID INT NOT NULL,

QUANTITY INT NOT NULL,

PRICE DECIMAL(8,2) NOT NULL,

FOREIGN KEY (INVOICE\_ID) REFERENCES INVOICE(INVOICE\_ID) ON DELETE CASCADE,

FOREIGN KEY (PRODUCT\_ID) REFERENCES PRODUCT(PRODUCT\_ID) ON DELETE CASCADE

);

The CUSTOMER and VENDOR tables store information about customers and vendors respectively. The PRODUCT table stores information about products sold by vendors. The INVOICE table stores information about each order placed by a customer, and the LINE table stores information about each product ordered as part of an invoice.

Note that the VENDOR\_ID column in the PRODUCT table is a foreign key referencing the VENDOR\_ID column in the VENDOR table, which enforces referential integrity. Similarly, the CUSTOMER\_ID column in the INVOICE table is a foreign key referencing the CUSTOMER\_ID column in the CUSTOMER table, and the INVOICE\_ID and PRODUCT\_ID columns in the LINE table are foreign keys referencing the INVOICE\_ID and PRODUCT\_ID columns in the INVOICE and PRODUCT tables, respectively.