Odne Rindheim

Task 1: DBMS statements and utilities

Creating database with NORMAL user:

CREATE DATABASE assignment2task1

USE assignment2task1

Creating 2 tables (I made 1 with InnoDB and 1 with MyISAM):

CREATE TABLE innodb (id INT AUTO_INCREMENT PRIMARY KEY, name VARCHAR(100)) ENGINE = InnoDB;

CREATE TABLE myisam (id INT AUTO_INCREMENT PRIMARY KEY, name VARCHAR(100) ENGINE = MyISAM;

Inserting Test Data:

INSERT INTO test_innodb (name) VALUES ('John'), ('Jane'), ('Doe');
INSERT INTO test_myisam (name) VALUES ('Alice'), ('Bob'), ('Charlie');

Converted the tables from InnoDB to MyISAM, and vice versa:

ALTER TABLE employees_innodb ENGINE=MyISAM;

ALTER TABLE employees_myisam ENGINE=InnoDB;

Showing index:

SHOW INDEX FROM employees_innodb;

SHOW INDEX FROM employees_myisam;

Why/when would you use this command

The ‘SHOW INDEX’ command is used to display the indexeds of a table, providing detailes about each index such as its name, type, columns it is composed of, and whether it is unique or not. This command is particularly useful when optimizing queries, understanding how data is accessed and organized, and ensuring that joins and searches are fficient. By reviewing the indexes, you can determine if additional indexes are needed to improve performance or if exisiting ones need to be modified to better serve the database queries.

Analyze table:

ANALYZE TABLE employees_innodb;

Odne Rindheim

ANALYZE TABLE employees_myisam;

Output: Op analyze; Msg_type status; Msg_text OK.

When would you use this command?

You would use the 'Analyze table' command to update the statistics for a table, which the mysql optimizer uses to choose the most efficient query execution plans. This command is particularly useful after significant changes to the table data, such as large inserts, updates, or deletes, which might alter the distribution of data within the table. Running 'analyze table' helps ensure that the mysql query optimizer has accurate information for making decisions about how to execute queries as efficiently as possible.

Check table:

CHECK TABLE employees_innodb;

CHECK TABLE employees_myisam;

Output: Op check; Msg_Type satus; Msg_text OK.

Why/when would you use this command? What can cause errors in tables?
The 'check table' command is crucial for verifying the integrity of mysql and mariadb tables, particularly useful after unexpected shutdowns, for regular maintenance, before and after backups, or when unusual database behaviors suggest potential issues. This command helps diagnose problems stemming from hardware failures, software bugs, improper shutdowns, malicious attacks, filesystem corruption, or network issues, ensuring tables are not corrupted and maintain data consistency. By identifying these issues early, 'check table' allows for timely repairs or restorations from backups, safeguarding against data loss and maintaining database reliability.

Repair Table:

REPAIR TABLE employees_innodb;

Output: Op repair; Msg_type status; Msg_text OK.

REPAIR TABLE employees_myisam;

Output: Op repair; Msg_type note; Msg_text The storage engine for the table doesn't support repair.

When can a table become corrupted? Why/when would you use this command?

A table can become corrupted due to several reasons, such as hardware failures (e.g., disk issues or power outages leading to incomplete writes), software crashes, bugs in the database software, or even due to issues arising from improper shutdowns of the database server. Other factors include filesystem corruption, running out of disk space during operations, or malicious activity that damages files.

The repair table command is used when you have identified that a table has become corrupted and needs to be fixed to restore its integrity and functionality. This command is specifically designed for repairing issues within tables, especially for storage engines like MyISAM, which do not automatically recover from corruption as some other storage engines might.

Optimize table:

OPTIMIZE TABLE employees_innodb;

Output: Op optimize; Msg_type status; Msg_text OK.

OPTIMIZE TABLE employees_myisam,

Output: Op optimize; Msg_type note; Msg_text Table does not support optimize, doing recreate+analyze instead.
Op optimize; Msg_type status; Msg_text OK

When do we need to optimize table?

Optimizing a table is necessary when you want to improve the efficiency of database operations by reclaiming unused space and defragmenting the data. This process becomes particularly important after significant changes have been made to a table, such as large numbers of rows being inserted, updated, or deleted. These operations can lead to fragmented data that occupies more space than necessary and slows down query performance. By running the optimize table command, you effectively compact the data, which can lead to faster data retrieval and more efficient space usage. Its also useful for tables that are accessed frequently or have undergone bulk changes to ensure that the database maintains optimal performance and uses disk space efficiently. Regularly scheduling table optimization as part of database maintenance can help in keeping the database responsive and in good health.

Checksum of tables:

CHECKSUM TABLE employees_innodb;

CHECHSUM TABLE employees_myisam;

Output: Checksum 1947335020.

Odne Rindheim
When can this be useful?

The checksum command can be incredible useful in scenarios where data integrity and consistency are paramount. This command computes a checksum for the contents of a table, providing a unique numerical value that represents the current state of the tables data. This deature is particularly valuable in enviroments where databases are replicated across multiple servers, as it allows database administrators to verify that tables are consistent across different nodes by comparing their checksum values. Its also useful after data migrations or backups to ensure that no corruption has occurred during the process, ensuring that the data before and after the operation is identical. Additionally, in troubleshooting scenarios where data corruption is suspected, the checksum can help conrifm if the data has indeed by altered. Employing checksum table as a routing part of data validation processes can significantly enhance data reliability and integrity checks within a database management system.

Innochecksum:

Sudo systemctl stop mariadb

Sudo innochecksum /var/lib/mysql/assignment2task1/employee_myisam.ibd

What can can cause problems with the integrity of InnoDB tablespace?
Problems with the integrity of innodb tablespaces can arise from hardware faillures, power outages, filesystem corruption, improper server shutdowns, software bugs, manual file manipulation, or external modifications. Ensuring regular backups, using reliable hardware, and maintaining updated software are critical to safeguarding data integrity

Myisamchk:

Sudo myisamchk /var/lib/mysql/assignment2task1/employee_innodb.MYI

Output: Data records : 2 Deleted blocks : 0

Sudo systemctl start mariadb

When can this be useful?

‘myisamchk’ is useful for checking, repairing, or optimizing MyISAM tables outside of the MySQL server environment. Its particularly valuable after a crash, for tables that wont open normally due to corruption, or when you need to improve performance by optimizing the table structure without accessing the MySQL server.

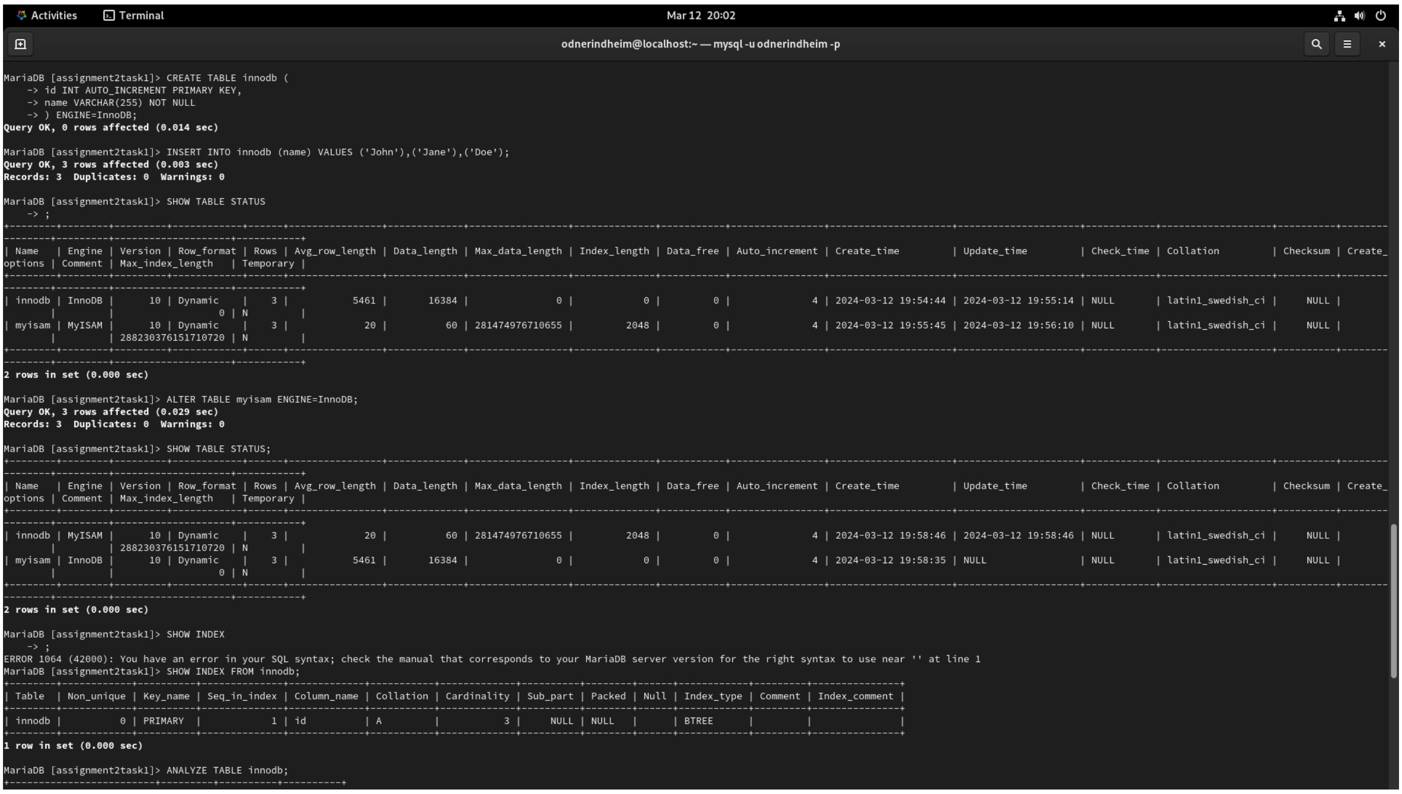
1. What would happen if you try REPAIR TABLE on the InnoDB Table (employees_myisam)?

Odne Rindheim

   a. REPAIR TABLE doesn't work for InnoDB tables because InnoDB has its own built-in crash recovery mechanism. Attempting to use it will result in an error message, and it doing recreate+analyze instead.

2. When would you use the innochecksum program?
   a. Can use innochecksum to verify the intergrity of InnoDB tablespace files (.idb) when the sql server isn't running.

Screenshots:



```
Activities   Terminal   Mar 12 20:02
odnerindheim@localhost:~ — mysql -u odnerindheim -p

MariaDB [assignment2task1]> CREATE TABLE innodb (
    -> id INT AUTO_INCREMENT PRIMARY KEY,
    -> name VARCHAR(255) NOT NULL
    -> ) ENGINE=InnoDB;
Query OK, 0 rows affected (0.014 sec)

MariaDB [assignment2task1]> INSERT INTO innodb (name) VALUES ('John'),('Jane'),('Doe');
Query OK, 3 rows affected (0.003 sec)
Records: 3  Duplicates: 0  Warnings: 0

MariaDB [assignment2task1]> SHOW TABLE STATUS
    -> ;
| Name   | Engine | Version | Row_format | Rows | Avg_row_length | Data_length | Max_data_length | Index_length | Data_free | Auto_increment | Create_time         | Update_time         | Check_time | Collation         | Checksum | Create_options | Comment | Max_index_length   | Temporary |
| innodb | InnoDB |      10 | Dynamic    |    3 |           5461 |       16384 |               0 |            0 |         0 |              4 | 2024-03-12 19:54:44 | 2024-03-12 19:55:14 | NULL       | latin1_swedish_ci |     NULL |                |         |                  0 | N         |
| myisam | MyISAM |      10 | Dynamic    |    3 |             20 |          60 | 281474976710655 |         2048 |         0 |              4 | 2024-03-12 19:55:45 | 2024-03-12 19:56:10 | NULL       | latin1_swedish_ci |     NULL |                |         | 288230376151710720 | N         |
2 rows in set (0.000 sec)

MariaDB [assignment2task1]> ALTER TABLE myisam ENGINE=InnoDB;
Query OK, 3 rows affected (0.029 sec)
Records: 3  Duplicates: 0  Warnings: 0

MariaDB [assignment2task1]> SHOW TABLE STATUS;
| Name   | Engine | Version | Row_format | Rows | Avg_row_length | Data_length | Max_data_length | Index_length | Data_free | Auto_increment | Create_time         | Update_time         | Check_time | Collation         | Checksum | Create_options | Comment | Max_index_length   | Temporary |
| innodb | MyISAM |      10 | Dynamic    |    3 |             20 |          60 | 281474976710655 |         2048 |         0 |              4 | 2024-03-12 19:58:46 | 2024-03-12 19:58:46 | NULL       | latin1_swedish_ci |     NULL |                |         | 288230376151710720 | N         |
| myisam | InnoDB |      10 | Dynamic    |    3 |           5461 |       16384 |               0 |            0 |         0 |              4 | 2024-03-12 19:58:35 | NULL                | NULL       | latin1_swedish_ci |     NULL |                |         |                  0 | N         |
2 rows in set (0.000 sec)

MariaDB [assignment2task1]> SHOW INDEX
    -> ;
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MariaDB server version for the right syntax to use near '' at line 1
MariaDB [assignment2task1]> SHOW INDEX FROM innodb;
| Table  | Non_unique | Key_name | Seq_in_index | Column_name | Collation | Cardinality | Sub_part | Packed | Null | Index_type | Comment | Index_comment |
| innodb |          0 | PRIMARY  |            1 | id          | A         |           3 |     NULL | NULL   |      | BTREE      |         |               |
1 row in set (0.000 sec)

MariaDB [assignment2task1]> ANALYZE TABLE innodb;
```

Odne Rindheim

Activities   Terminal   Mar 12 20:02

odnerindheim@localhost:~ — mysql -u odnerindheim -p

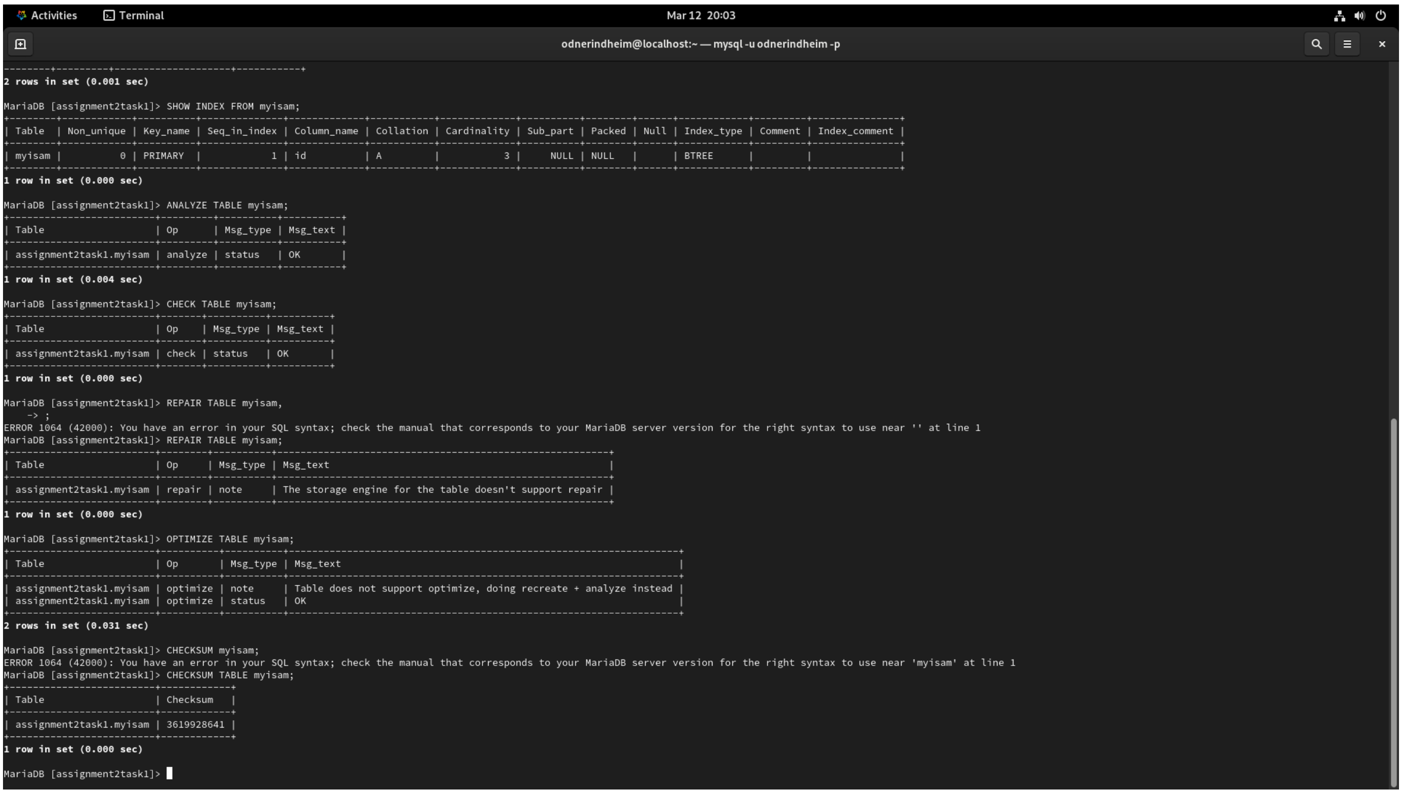
```
| myisam | InnoDB |      10 | Dynamic    |    3 |           5461 |       16384 |               0 |            0 |         0 |              4 | 2024-03-12 19:58:35 | NULL                | NULL       | latin1_swedish_ci |     NULL |
        |         |                   0 | N         |
+--------+--------+---------+------------+------+----------------+-------------+-----------------+--------------+-----------+----------------+---------------------+---------------------+------------+-------------------+----------+--------
--------+---------+--------------------+-----------+
2 rows in set (0.000 sec)

MariaDB [assignment2task1]> SHOW INDEX
    -> ;
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MariaDB server version for the right syntax to use near '' at line 1
MariaDB [assignment2task1]> SHOW INDEX FROM innodb;
+--------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+
| Table  | Non_unique | Key_name | Seq_in_index | Column_name | Collation | Cardinality | Sub_part | Packed | Null | Index_type | Comment | Index_comment |
+--------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+
| innodb |          0 | PRIMARY  |            1 | id          | A         |           3 |     NULL | NULL   |      | BTREE      |         |               |
+--------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+
1 row in set (0.000 sec)

MariaDB [assignment2task1]> ANALYZE TABLE innodb;
+-------------------------+---------+----------+----------+
| Table                   | Op      | Msg_type | Msg_text |
+-------------------------+---------+----------+----------+
| assignment2task1.innodb | analyze | status   | OK       |
+-------------------------+---------+----------+----------+
1 row in set (0.000 sec)

MariaDB [assignment2task1]> CHECK TABLE innodb;
+-------------------------+-------+----------+----------+
| Table                   | Op    | Msg_type | Msg_text |
+-------------------------+-------+----------+----------+
| assignment2task1.innodb | check | status   | OK       |
+-------------------------+-------+----------+----------+
1 row in set (0.000 sec)

MariaDB [assignment2task1]> REPAIR TABLE innodb;
+-------------------------+--------+----------+----------+
| Table                   | Op     | Msg_type | Msg_text |
+-------------------------+--------+----------+----------+
| assignment2task1.innodb | repair | status   | OK       |
+-------------------------+--------+----------+----------+
1 row in set (0.000 sec)

MariaDB [assignment2task1]> OPTIMIZE TABLE innodb;
+-------------------------+----------+----------+----------+
| Table                   | Op       | Msg_type | Msg_text |
+-------------------------+----------+----------+----------+
| assignment2task1.innodb | optimize | status   | OK       |
+-------------------------+----------+----------+----------+
1 row in set (0.000 sec)

MariaDB [assignment2task1]> CHECKSUM TABLE innodb;
+-------------------------+------------+
| Table                   | Checksum   |
+-------------------------+------------+
| assignment2task1.innodb | 2051758887 |
+-------------------------+------------+
1 row in set (0.000 sec)

MariaDB [assignment2task1]> 
```

Activities   Terminal   Mar 12 20:02

odnerindheim@localhost:~ — mysql -u odnerindheim -p

```
MariaDB [assignment2task1]>
MariaDB [assignment2task1]> CREATE TABLE myisam (
    -> id INT AUTO_INCREMENT PRIMARY KEY,
    -> name VARCHAR(255) NOT NULL
    -> ) ENGINE = MyISAM;
Query OK, 0 rows affected (0.005 sec)

MariaDB [assignment2task1]> INSERT INTO myisam (name) VALUES ('Alice'), ('Bob'),('Charlie');
Query OK, 3 rows affected (0.000 sec)
Records: 3  Duplicates: 0  Warnings: 0

MariaDB [assignment2task1]> SHOW TABLE STATUS
    -> ;
+--------+--------+---------+------------+------+----------------+-------------+-----------------+--------------+-----------+----------------+---------------------+---------------------+------------+-------------------+----------+--------
--------+---------+--------------------+-----------+
| Name   | Engine | Version | Row_format | Rows | Avg_row_length | Data_length | Max_data_length | Index_length | Data_free | Auto_increment | Create_time         | Update_time         | Check_time | Collation         | Checksum | Create_
options | Comment | Max_index_length   | Temporary |
+--------+--------+---------+------------+------+----------------+-------------+-----------------+--------------+-----------+----------------+---------------------+---------------------+------------+-------------------+----------+--------
--------+---------+--------------------+-----------+
| innodb | InnoDB |      10 | Dynamic    |    3 |           5461 |       16384 |               0 |            0 |         0 |              4 | 2024-03-12 19:54:44 | 2024-03-12 19:55:14 | NULL       | latin1_swedish_ci |     NULL |
        |         |                   0 | N         |
| myisam | MyISAM |      10 | Dynamic    |    3 |             20 |          60 | 281474976710655 |         2048 |         0 |              4 | 2024-03-12 19:55:45 | 2024-03-12 19:56:10 | NULL       | latin1_swedish_ci |     NULL |
        |         | 288230376151710720 | N         |
+--------+--------+---------+------------+------+----------------+-------------+-----------------+--------------+-----------+----------------+---------------------+---------------------+------------+-------------------+----------+--------
--------+---------+--------------------+-----------+
2 rows in set (0.000 sec)

MariaDB [assignment2task1]> ALTER TABLE innodb ENGINE=MyISAM;
Query OK, 3 rows affected (0.016 sec)
Records: 3  Duplicates: 0  Warnings: 0

MariaDB [assignment2task1]> SHOW TABLE STATUS;
+--------+--------+---------+------------+------+----------------+-------------+-----------------+--------------+-----------+----------------+---------------------+---------------------+------------+-------------------+----------+--------
--------+---------+--------------------+-----------+
| Name   | Engine | Version | Row_format | Rows | Avg_row_length | Data_length | Max_data_length | Index_length | Data_free | Auto_increment | Create_time         | Update_time         | Check_time | Collation         | Checksum | Create_
options | Comment | Max_index_length   | Temporary |
+--------+--------+---------+------------+------+----------------+-------------+-----------------+--------------+-----------+----------------+---------------------+---------------------+------------+-------------------+----------+--------
--------+---------+--------------------+-----------+
| innodb | MyISAM |      10 | Dynamic    |    3 |             20 |          60 | 281474976710655 |         2048 |         0 |              4 | 2024-03-12 19:58:46 | 2024-03-12 19:58:46 | NULL       | latin1_swedish_ci |     NULL |
        |         | 288230376151710720 | N         |
| myisam | InnoDB |      10 | Dynamic    |    3 |           5461 |       16384 |               0 |            0 |         0 |              4 | 2024-03-12 19:58:35 | NULL                | NULL       | latin1_swedish_ci |     NULL |
        |         |                   0 | N         |
+--------+--------+---------+------------+------+----------------+-------------+-----------------+--------------+-----------+----------------+---------------------+---------------------+------------+-------------------+----------+--------
--------+---------+--------------------+-----------+
2 rows in set (0.001 sec)

MariaDB [assignment2task1]> SHOW INDEX FROM myisam;
+--------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+
| Table  | Non_unique | Key_name | Seq_in_index | Column_name | Collation | Cardinality | Sub_part | Packed | Null | Index_type | Comment | Index_comment |
+--------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+
| myisam |          0 | PRIMARY  |            1 | id          | A         |           3 |     NULL | NULL   |      | BTREE      |         |               |
+--------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+
1 row in set (0.000 sec)

MariaDB [assignment2task1]> ANALYZE TABLE myisam;
+-------------------------+---------+----------+----------+
| Table                   | Op      | Msg_type | Msg_text |
+-------------------------+---------+----------+----------+
| assignment2task1.myisam | analyze | status   | OK       |
```

Odne Rindheim




```
--------+----------+--------------------+-----------+
2 rows in set (0.001 sec)

MariaDB [assignment2task1]> SHOW INDEX FROM myisam;
+--------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+
| Table  | Non_unique | Key_name | Seq_in_index | Column_name | Collation | Cardinality | Sub_part | Packed | Null | Index_type | Comment | Index_comment |
+--------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+
| myisam |          0 | PRIMARY  |            1 | id          | A         |           3 |     NULL | NULL   |      | BTREE      |         |               |
+--------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+
1 row in set (0.000 sec)

MariaDB [assignment2task1]> ANALYZE TABLE myisam;
+-------------------------+---------+----------+----------+
| Table                   | Op      | Msg_type | Msg_text |
+-------------------------+---------+----------+----------+
| assignment2task1.myisam | analyze | status   | OK       |
+-------------------------+---------+----------+----------+
1 row in set (0.004 sec)

MariaDB [assignment2task1]> CHECK TABLE myisam;
+-------------------------+-------+----------+----------+
| Table                   | Op    | Msg_type | Msg_text |
+-------------------------+-------+----------+----------+
| assignment2task1.myisam | check | status   | OK       |
+-------------------------+-------+----------+----------+
1 row in set (0.000 sec)

MariaDB [assignment2task1]> REPAIR TABLE myisam,
    -> ;
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MariaDB server version for the right syntax to use near '' at line 1
MariaDB [assignment2task1]> REPAIR TABLE myisam;
+-------------------------+--------+----------+---------------------------------------------------------+
| Table                   | Op     | Msg_type | Msg_text                                                |
+-------------------------+--------+----------+---------------------------------------------------------+
| assignment2task1.myisam | repair | note     | The storage engine for the table doesn't support repair |
+-------------------------+--------+----------+---------------------------------------------------------+
1 row in set (0.000 sec)

MariaDB [assignment2task1]> OPTIMIZE TABLE myisam;
+-------------------------+----------+----------+-------------------------------------------------------------------+
| Table                   | Op       | Msg_type | Msg_text                                                          |
+-------------------------+----------+----------+-------------------------------------------------------------------+
| assignment2task1.myisam | optimize | note     | Table does not support optimize, doing recreate + analyze instead |
| assignment2task1.myisam | optimize | status   | OK                                                                |
+-------------------------+----------+----------+-------------------------------------------------------------------+
2 rows in set (0.031 sec)

MariaDB [assignment2task1]> CHECKSUM myisam;
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MariaDB server version for the right syntax to use near 'myisam' at line 1
MariaDB [assignment2task1]> CHECKSUM TABLE myisam;
+-------------------------+------------+
| Table                   | Checksum   |
+-------------------------+------------+
| assignment2task1.myisam | 3619928641 |
+-------------------------+------------+
1 row in set (0.000 sec)

MariaDB [assignment2task1]> 
```

Odne Rindheim

Task 2: Normalization

1. What normal form does it currently conform to? Normalize it to 3NF I it does not currently conform to 3NF.
    a. The given table is 1NF because each attribute contains only atomic values, and there are no repeating groups.
    It is not 2NF because there is partial dependency. The employee's name (ENAME) is dependent on SSN, and not on the composite key (SSN, PNUMBER). Same for project name (PNAME) and location (PLOCATION) are dependent on the PNUMBER, and not the composite key.
    It is not 3NF because there are transitive dependencies present. Employee name (ENAME) is dependent on SSN, which is not the primary key. The same goes for PNAME and PLOCATION which are dependent on PNUMBER.
    b. To normalize the relation to 3NF, we need to remove the partial and transitive dependencies by creating separate tables.
    This normalized design removes the partial and transitive dependencies, and each non-key attribute is only dependent on the key. The EMP_PROJ table maintains the association between EMPLOYEE and PROJECT, with HOURS indicating the number hours an employee has worked on a project.

EMPLOYEE	**PROJECT**	**EMP_PROJ**
SSN (PK)	PNUMBER (PK)	SSN (Composite key, FK)
ENAME	PNAME	PNUMBER (CK, FK)
	PLOCATION	HO

Odne Rindheim

Task 3: More normalization

1. Which normal form does it currently conform to? Normalize it to 3NF if it does not currently conform to 3NF.
    a. As all data is atomic, its atleast in 1nf. Since all data is linked to the primary key, we are also in 2nf. But since DNAME and DMGRSSN are linked transitively to the key through DNUMBER, we split these out of the table and create a new one, thus the model will now be:

EMPLOYEE	**DEPARTMENT**
SSN (PK)	DNUMBER(PK)
ENAME	DNAME
BDATE	DMGRSSN
ADDRESS	
DNUMBER (FK)	

It's now in 3nf since all non-key attributes are now non-transitively bound to the primary key of the table.

Odne Rindheim

Task 4: Normalization & Denormalization

1. Creating the scheme

STUDENT	FACULTY	DEPARTMENT	COURSE	COURSE_SCHEDULE	GRADE	TEACHER
Student_Number (PK)	FCode (PK)	Dept_ID (PK)	Course_Code (PK)	CSchedule_ID (PK)	Grade_ID (PK)	Teacher_ID (PK)
SNAME	FName	Dept_Name	Course_Name	CourseCode (FK)	Student_Number (FK)	Teacher_Name
Birth_Number	Phone	FCode (FK)	LectureHours	TeacherID (FK)	CSchedule_ID (FK)	Dept_ID (FK)
Address	Address		Dept_ID (FK)	CYEAR	Grade	
Phone						
Gender						
Year						
FCode (FK)						
Study_Program						
Study_Level						

2. Database scheme. // updated with PK course shceudle. Varchars could've been char to save space and performance.

CREATE TABLE FACULTY (

FCODE VARCHAR(10) PRIMARY KEY,

FNAME VARCHAR(255) NOT NULL UNIQUE,

PHONE VARCHAR(20),

ADDRESS VARCHAR(255)

);

CREATE TABLE DEPARTMENT (

Dept_ID INT AUTO_INCREMENT PRIMARY KEY,

Dept_Name VARCHAR(255) NOT NULL,

FCODE VARCHAR(10),

FOREIGN KEY (FCODE) REFERENCES FACULTY(FCODE)

);

CREATE TABLE STUDENT (

Student_Number VARCHAR(7) PRIMARY KEY,

Sname VARCHAR(255) NOT NULL,

Birth_Number VARCHAR(11) UNIQUE NOT NULL,

Address VARCHAR(255),

```
Phone VARCHAR(20),
Gender enum(“M”,”F”,”Other”),
Year INT,
FCODE VARCHAR(10),
Study_Program VARCHAR(100),
Study_Level enum('Bachelor','Master','PHD'),
FOREIGN KEY (Faculty_Code) REFERENCES FACULTY(FCODE)
);

CREATE TABLE COURSE (
Course_Code VARCHAR(20) PRIMARY KEY,
Course_Name VARCHAR(255) NOT NULL,
Lecture_Hours INT NOT NULL,
Dept_ID INT,
FOREIGN KEY (Dept_ID) REFERENCES DEPARTMENT(Dept_ID)
);

CREATE TABLE TEACHER (
Teacher_ID INT AUTO_INCREMENT PRIMARY KEY,
Teacher_Name VARCHAR(255) NOT NULL,
Dept_ID INT,
FOREIGN KEY (Dept_ID) REFERENCES DEPARTMENT(Dept_ID)
);

CREATE TABLE COURSE_SCHEDULE (
Cschedule_ID INT AUTO_INCREMENT PRIMARY KEY,
Course_Code VARCHAR(20),
Teacher_ID INT,
CYEAR YEAR,
FOREIGN KEY (Course_Code) REFERENCES COURSE(Course_Code),
FOREIGN KEY (Teacher_ID) REFERENCES TEACHER(Teacher_ID)
);

CREATE TABLE GRADE (
```

```
Grade_ID INT AUTO_INCREMENT PRIMARY KEY,
Student_Number VARCHAR(7),
Cschedule_ID INT,
GRADE VARCHAR(5),
FOREIGN KEY (Student_Number) REFERENCES STUDENT (Student_Number),
FOREIGN KEY (Cschedule_ID) REFERENCES COURSE_SCHEDULE(Cschedule_ID),
UNIQUE (Student_Number, Cschedule_ID)
);
```

3. SELECT s.NAME, f.FNAME FROMT STUDENT s, FACTULTY f WHERE s.FCODe = f.CODE;
    a. Denormalization, by incorporating the factulty name into the student table, can notably streamline this query by obviating the need for a join operation between student and faculty tables. This modification would directly associate students with their faculties, enhancing read performance for queries that necessitate both student and factulty information. Nevertheless, this approach introduces a potential challenge for database maintenance, as any modifications to faculty names would necessitate update across multiple rows within the student table. This could lead to a higher risk of data inconsistency and increased workload for write operations, especially in a dynamic environment where faculty details are prone to change. Hence, while denormalization may enhance query efficiency, it requires a balanced consideration of the impact on data integrity and the additional overhead for maintaining consistency.
4. SELECT COUNT (*) FROM STUDENT WHERE FCODE = 'FIN';
    a. With this query denormalization could offer performance improvements. While the query itself is inherently efficient due to its reliance on an indexed attribute (FCODE), denormalization could still offer advantages, particularly in scenarios with high concurrency and potential lock issues. By denormalizing the schema and adding a column to the 'FACULTY' table, such as 'STUDENT_COUNT', which stores the count of students associated with each faculty, the need for the COUNT operation in the query can be eliminated. This precomputed denormalized value provides an instant retrieval of the student count for a specific faculty without requiring and expensive COUNT operation on the STUDENT table. When a new student is added or removed from a faculty, the 'STUDENT_COUNT' column is updated accordingly. The query can be rewritten as 'SELECT STUDENT_COUNT FROM FACULTY WHERE FCODE = 'FIN';', which retrieves the precomputed count directly from the faculty table, avoiding the need for

counting records on the fly. This approach eliminates the need for COUNT, but also reduces contention on the 'STUDENT' table, alleviating potential concurrency issues and locks.

5. SELECT DISTINCT CYEAR FROM COURSE_SCHEDULE
    a. In tis query, the focus is on retrieving distinct years from the 'Course_schedule' table. While indexing strategies can certainly enhance performance by facilitating rapid retrieval of distinct values, denormalization could potentially offer improvements in scenarios with high concurrency and lock issues. One possible denormalization approach involves precomputing and storing distinct years in a separate table or adding a column to an existing table that maintains a list of distinct years. This eliminates the need for DISTINCT operation, improving query performance, especially in situations with frequent concurrent transactions and lock contention. For example, we create a denormalized table named 'Distinct_years' with a single column 'CYEAR' whenever a new course schedule is added or updated, the distinct years are automatically updated in the 'Distinct_years' table. Then, the query can be writted as 'SELECT CYEAR FROM DISTINCT_YEARS;' which directly retries the precomputed distinct year without need for distinct operation in course_scheudle table. This approach reduces computational overhead of distinct operation and mitigate concurrency issues associated with locking, enhancing overall performance and scalability of dbms.