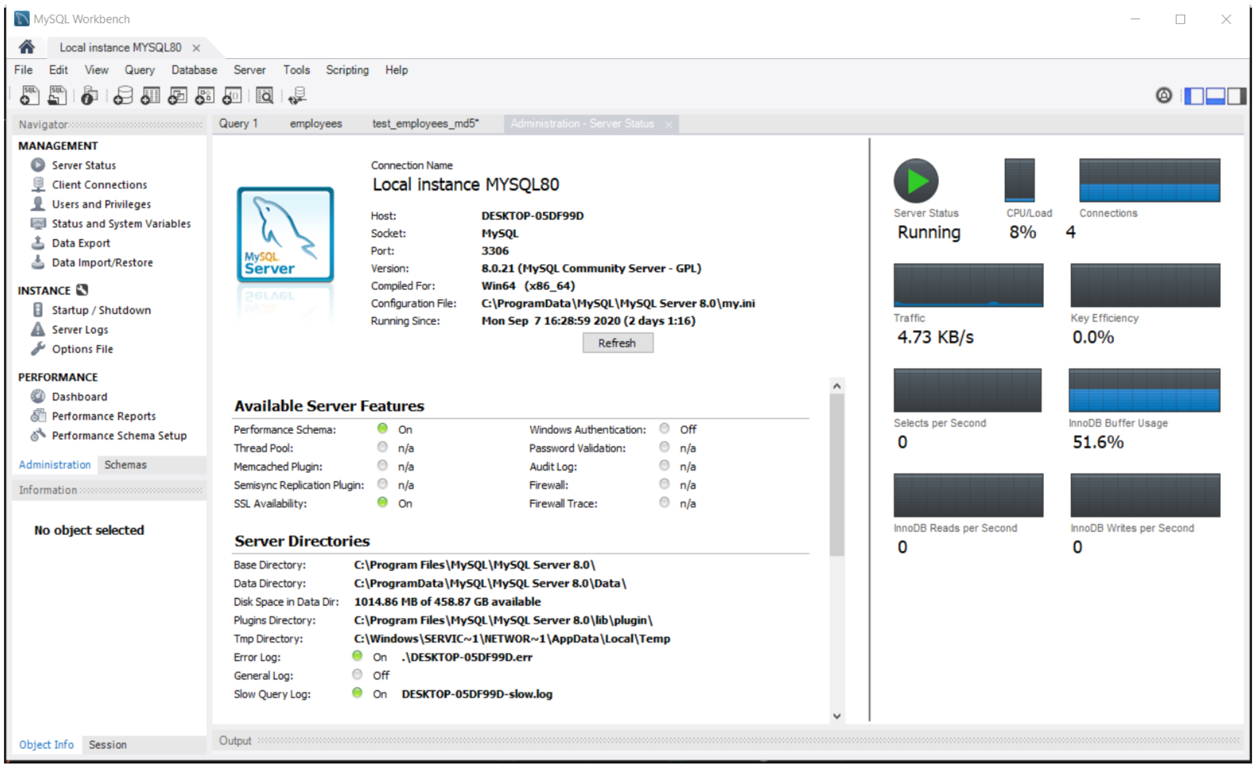
CSE 4/560 PA 0: Hello SQorLd

NAME: ASHISH SANGHVI

UBIT NAME: ashishsa

UBIT NUMBER: 50318479



Connection Name
Local instance MYSQL80

Host:	DESKTOP-05DF99D
Socket:	MySQL
Port:	3306
Version:	8.0.21 (MySQL Community Server - GPL)
Compiled For:	Win64 (x86_64)
Configuration File:	C:\ProgramData\MySQL\MySQL Server 8.0\my.ini
Running Since:	Mon Sep 7 16:28:59 2020 (2 days 1:16)

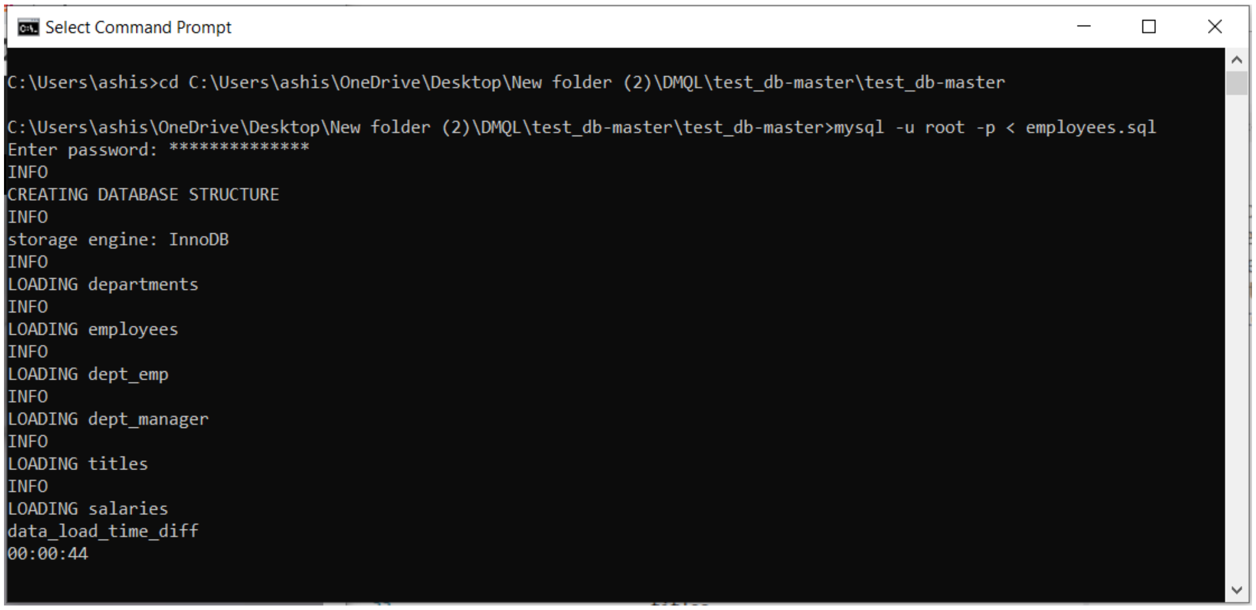
Server Status: Running · CPU/Load: 8% · Connections: 4 · Traffic: 4.73 KB/s · Key Efficiency: 0.0% · Selects per Second: 0 · InnoDB Buffer Usage: 51.6% · InnoDB Reads per Second: 0 · InnoDB Writes per Second: 0

Available Server Features

Performance Schema:	On	Windows Authentication:	Off
Thread Pool:	n/a	Password Validation:	n/a
Memcached Plugin:	n/a	Audit Log:	n/a
Semisync Replication Plugin:	n/a	Firewall:	n/a
SSL Availability:	On	Firewall Trace:	n/a

Server Directories

Base Directory:	C:\Program Files\MySQL\MySQL Server 8.0\
Data Directory:	C:\ProgramData\MySQL\MySQL Server 8.0\Data\
Disk Space in Data Dir:	1014.86 MB of 458.87 GB available
Plugins Directory:	C:\Program Files\MySQL\MySQL Server 8.0\lib\plugin\
Tmp Directory:	C:\Windows\SERVIC~1\NETWOR~1\AppData\Local\Temp
Error Log:	On .\DESKTOP-05DF99D.err
General Log:	Off
Slow Query Log:	On DESKTOP-05DF99D-slow.log



```
C:\Users\ashis>cd C:\Users\ashis\OneDrive\Desktop\New folder (2)\DMQL\test_db-master\test_db-master

C:\Users\ashis\OneDrive\Desktop\New folder (2)\DMQL\test_db-master\test_db-master>mysql -u root -p < employees.sql
Enter password: **************
INFO
CREATING DATABASE STRUCTURE
INFO
storage engine: InnoDB
INFO
LOADING departments
INFO
LOADING employees
INFO
LOADING dept_emp
INFO
LOADING dept_manager
INFO
LOADING titles
INFO
LOADING salaries
data_load_time_diff
00:00:44
```

IN STEP1: WE LOAD EMPLOYEES.SQL

In Employees.sql we have the following commands:

```
DROP DATABASE IF EXISTS employees;
```

WE DELETE THE EMPLOYEES DATABASE IF IT EXISTS.

```
CREATE DATABASE IF NOT EXISTS employees;
```

WE CREATE EMPLOYEES DATABASE.

```
USE employees;
```

AFTER CREATING EMPLOYEES DATABASE, WE USE EMPLOYEES DATABASE.

```
SELECT 'CREATING DATABASE STRUCTURE' as 'INFO';
```

RETREIVE INFORMATION FROM CREATING DATABASE STRUCTURE AS INFORMATION.

```
DROP TABLE IF EXISTS dept_emp,

                dept_manager,

                titles,

                salaries,

                employees,

                departments;
```

WE DELETE THE TABLES (DEPT_EMP, DEPT_MANAGER, TITLES, SALARIES, EMPLOYEES, DEPARTMENT) IF THEY ARE PRE-EXISTING WITH THE GIVEN NAME.

```
/*!50503 set default_storage_engine = InnoDB */;
```

WE SET INNODB AS THE DEFAULT SEARCH ENGINE.

```
/*!50503 select CONCAT('storage engine: ', @@default_storage_engine) as INFO */;
```

WE RETREIVE INFORMATION ON STORAGE ENGINE AND DEFAULT STORAGE ENGINE AND DISPLAY THEM AS INFO.

```
CREATE TABLE employees (

   emp_no      INT             NOT NULL,

   birth_date  DATE            NOT NULL,

   first_name  VARCHAR(14)     NOT NULL,

   last_name   VARCHAR(16)     NOT NULL,

   gender      ENUM ('M','F')  NOT NULL,

   hire_date   DATE            NOT NULL,

   PRIMARY KEY (emp_no)

);
```

WE CREATE EMPLOYEES TABLE WITH THE FOLLOWING FIELDS:

EMP_NO WHICH IS OF TYPE INTEGER AND DOES NOT ACCEPT NULL VALUES,

BIRTH_DATE WHICH IS OF TYPE DATE AND DOES NOT ACCEPT NULL VALUES,

FIRST_NAME WHICH IS OF TYPE VARCHAR AND DOES NOT ACCEPT NULL VALUES,

LAST_NAME WHICH IS OF TYPE VARCHAR AND DOES NOT ACCEPT NULL VALUES,

GENDER WHICH IS OF TYPE ENUM VALUES AND DOES NOT ACCEPT NULL VALUES,

HIRE_DATE WHICH IS OF TYPE DATE AND DOES NOT ACCEPT NULL VALUES,

SET EMP_NO AS THE PRIMARY KEY FOR THE TABLE.

```
CREATE TABLE departments (
   dept_no     CHAR(4)        NOT NULL,
   dept_name   VARCHAR(40)    NOT NULL,
   PRIMARY KEY (dept_no),
   UNIQUE  KEY (dept_name)
);
```

WE CREATE DEPARTMENTS TABLE WITH THE FOLLOWING FIELDS:

DEPT_NO WHICH IS OF TYPE CHAR AND DOES NOT ACCEPT NULL VALUES,

DEPT_NAME WHICH IS OF TYPE VARCHAR AND DOES NOT ACCEPT NULL VALUES,

SET DEPT_NO AS THE PRIMARY KEY FOR THE TABLE,

SET DEPT_NAME TO HAVE ONLY UNIQUE VALUES IN IT. (USER HAS TO ONLY ENTER UNIQUE VALUE, REPEATED ENTRY IS NOT ALLOWED).

```
CREATE TABLE dept_manager (
  emp_no      INT           NOT NULL,
  dept_no     CHAR(4)       NOT NULL,
  from_date   DATE          NOT NULL,
  to_date     DATE          NOT NULL,
  FOREIGN KEY (emp_no)  REFERENCES employees (emp_no)    ON DELETE CASCADE,
  FOREIGN KEY (dept_no) REFERENCES departments (dept_no) ON DELETE CASCADE,
  PRIMARY KEY (emp_no,dept_no)
);
```

WE CREATE DEPARTMENT_MANAGER TABLE WITH THE FOLLOWING FIELDS:

EMP_NO WHICH IS OF TYPE INTEGER AND DOES NOT ACCEPT NULL VALUES,

DEPT_NO WHICH IS OF TYPE CHAR AND DOES NOT ACCEPT NULL VALUES,

FROM_DATE WHICH IS OF TYPE DATE AND DOES NOT ACCEPT NULL VALUES,

TO_DATE WHICH IS OF TYPE DATE AND DOES NOT ACCEPT NULL VALUES,

SET EMP_NO AS THE FOREIGN KEY FOR THE TABLE WHOSE VALUE IS REFERENCED FROM THE (EMP_NO COLUMN) OF EMPLOYEES TABLE AND IS SET TO DELETE FROM RECORDS IF THE PARENT ELEMENT IS DELETED FROM EMPLOYEES TABLE (ON DELETE CASCADE).

SET DEPT_NO AS THE FOREIGN KEY FOR THE TABLE WHOSE VALUE IS REFERENCED FROM THE (DEPT_NO COLUMN) OF DEPARTMENTS TABLE AND IS SET TO DELETE FROM RECORDS IF THE PARENT ELEMENT IS DELETED FROM DEPARTMENT TABLE (ON DELETE CASCADE).

WE ALSO SET EMP_NO AND DEPT_NO AS THE PRIMARY KEY.

```
CREATE TABLE dept_emp (

   emp_no      INT           NOT NULL,

   dept_no     CHAR(4)        NOT NULL,

   from_date   DATE           NOT NULL,

   to_date     DATE           NOT NULL,

   FOREIGN KEY (emp_no)  REFERENCES employees   (emp_no)  ON DELETE CASCADE,

   FOREIGN KEY (dept_no) REFERENCES departments (dept_no) ON DELETE CASCADE,

   PRIMARY KEY (emp_no,dept_no)

);
```

WE CREATE DEPARTMENT_EMPLOYEE TABLE WITH THE FOLLOWING FIELDS:

EMP_NO WHICH IS OF TYPE INTEGER AND DOES NOT ACCEPT NULL VALUES,

DEPT_NO WHICH IS OF TYPE CHAR AND DOES NOT ACCEPT NULL VALUES,

FROM_DATE WHICH IS OF TYPE DATE AND DOES NOT ACCEPT NULL VALUES,

TO_DATE WHICH IS OF TYPE DATE AND DOES NOT ACCEPT NULL VALUES,

SET EMP_NO AS THE FOREIGN KEY FOR THE TABLE WHOSE VALUE IS REFERENCED FROM THE (EMP_NO COLUMN) OF EMPLOYEES TABLE AND IS SET TO DELETE FROM RECORDS IF THE PARENT ELEMENT IS DELETED FROM EMPLOYEES TABLE (ON DELETE CASCADE).

SET DEPT_NO AS THE FOREIGN KEY FOR THE TABLE WHOSE VALUE IS REFERENCED FROM THE (DEPT_NO COLUMN) OF DEPARTMENTS TABLE AND IS SET TO DELETE FROM RECORDS IF THE PARENT ELEMENT IS DELETED FROM DEPARTMENT TABLE (ON DELETE CASCADE).

WE ALSO SET EMP_NO AND DEPT_NO AS THE PRIMARY KEY.

```
CREATE TABLE titles (

   emp_no      INT           NOT NULL,

   title       VARCHAR(50)    NOT NULL,

   from_date   DATE           NOT NULL,

   to_date     DATE,

   FOREIGN KEY (emp_no) REFERENCES employees (emp_no) ON DELETE CASCADE,

   PRIMARY KEY (emp_no,title, from_date)

) ;
```

WE CREATE TITLES TABLE WITH THE FOLLOWING FIELDS:

EMP_NO WHICH IS OF TYPE INTEGER AND DOES NOT ACCEPT NULL VALUES,

TITLE WHICH IS OF TYPE VARCHAR AND DOES NOT ACCEPT NULL VALUES,

FROM_DATE WHICH IS OF TYPE DATE AND DOES NOT ACCEPT NULL VALUES,

TO_DATE WHICH IS OF TYPE DATE,

SET EMP_NO AS THE FOREIGN KEY FOR THE TABLE WHOSE VALUE IS REFERENCED FROM THE (EMP_NO COLUMN) OF EMPLOYEES TABLE AND IS SET TO DELETE FROM RECORDS IF THE PARENT ELEMENT IS DELETED FROM EMPLOYEES TABLE (ON DELETE CASCADE).

WE ALSO SET EMP_NO, TITLE, FROM_DATE AS THE PRIMARY KEY.

```
CREATE TABLE salaries (

   emp_no      INT            NOT NULL,

   salary      INT            NOT NULL,

   from_date   DATE           NOT NULL,

   to_date     DATE           NOT NULL,

   FOREIGN KEY (emp_no) REFERENCES employees (emp_no) ON DELETE CASCADE,

   PRIMARY KEY (emp_no, from_date)

) ;
```

WE CREATE SALARIES TABLE WITH THE FOLLOWING FIELDS:

EMP_NO WHICH IS OF TYPE INTEGER AND DOES NOT ACCEPT NULL VALUES,

SALARY WHICH IS OF TYPE INTEGER AND DOES NOT ACCEPT NULL VALUES,

FROM_DATE WHICH IS OF TYPE DATE AND DOES NOT ACCEPT NULL VALUES,

TO_DATE WHICH IS OF TYPE DATE AND DOES NOT ACCEPT NULL VALUES,

SET EMP_NO AS THE FOREIGN KEY FOR THE TABLE WHOSE VALUE IS REFERENCED FROM THE (EMP_NO COLUMN) OF EMPLOYEES TABLE AND IS SET TO DELETE FROM RECORDS IF THE PARENT ELEMENT IS DELETED FROM EMPLOYEES TABLE (ON DELETE CASCADE).

WE ALSO SET EMP_NO, FROM_DATE AS THE PRIMARY KEY.

```
CREATE OR REPLACE VIEW dept_emp_latest_date AS

   SELECT emp_no, MAX(from_date) AS from_date, MAX(to_date) AS to_date

   FROM dept_emp

   GROUP BY emp_no;
```

WE CREATE A VIRTUAL TABLE FROM DEPT_EMP TABLE USING THE FOLLOWING COMMAND (CREATE OR REPLACE VIEW) AND SELECT THE EMPLOYEE_NO OF EMPLOYEES WITH THE MAXIMUM FROM DATE AND TO DATES. (IE., EMPLOYEES WHICH HAVE BEEN EMPLOYEED FOR THE MAXIMUM TIME) AND GROUPING THEM BY THE EMP_NO.

```
# shows only the current department for each employee

CREATE OR REPLACE VIEW current_dept_emp AS

   SELECT l.emp_no, dept_no, l.from_date, l.to_date

   FROM dept_emp d
```

```
    INNER JOIN dept_emp_latest_date l

    ON d.emp_no=l.emp_no AND d.from_date=l.from_date AND l.to_date = d.to_date;
```

WE CREATE A VIRTUAL TABLE CALLED CURRENT_DEPT_EMP FROM DEPT_EMP TABLE USING THE FOLLOWING COMMAND (CREATE OR REPLACE VIEW) AND SELECTING THE EMPLOYEE_NO, DEPT_NO, FROM AND TO DATE OF EMPLOYEES WITH THE MAXIMUM FROM DATE AND TO DATES (IE., EMPLOYEES WHICH HAVE BEEN EMPLOYEED FOR THE MAXIMUM TIME) FROM DEPT_EMP TABLE AND INNER JOINING THEM WITH THE DEPT_EMP_LATEST_DATE ON (EMP_NO, FROM_DATE AND TO_DATE).

```
flush /*!50503 binary */ logs;
```

THE COMMAND CLOSES AND REOPENS THE LOG FILES

```
SELECT 'LOADING departments' as 'INFO'

source load_departments.dump ;
```

WE LOAD THE DEPARETMENTS AS INFO AND ALSO THE DUMP FILE FROM THE LOAD_DEPARTMENTS.

```
SELECT 'LOADING employees' as 'INFO';

source load_employees.dump ;
```

WE LOAD THE EMPLOYEES AS INFO AND ALSO THE DUMP FILE FROM THE LOAD_EMPLOYEES.

```
SELECT 'LOADING dept_emp' as 'INFO';

source load_dept_emp.dump ;
```

WE LOAD THE DEPT_EMP AS INFO AND ALSO THE DUMP FILE FROM THE LOAD_DEPARTMENTS_EMPLOYEE.

```
SELECT 'LOADING dept_manager' as 'INFO';

source load_dept_manager.dump ;
```

WE LOAD THE DEPARETMENTS_MANAGER AS INFO AND ALSO THE DUMP FILE FROM THE LOAD_DEPARTMENTS_MANAGER.

```
SELECT 'LOADING titles' as 'INFO';

source load_titles.dump ;
```

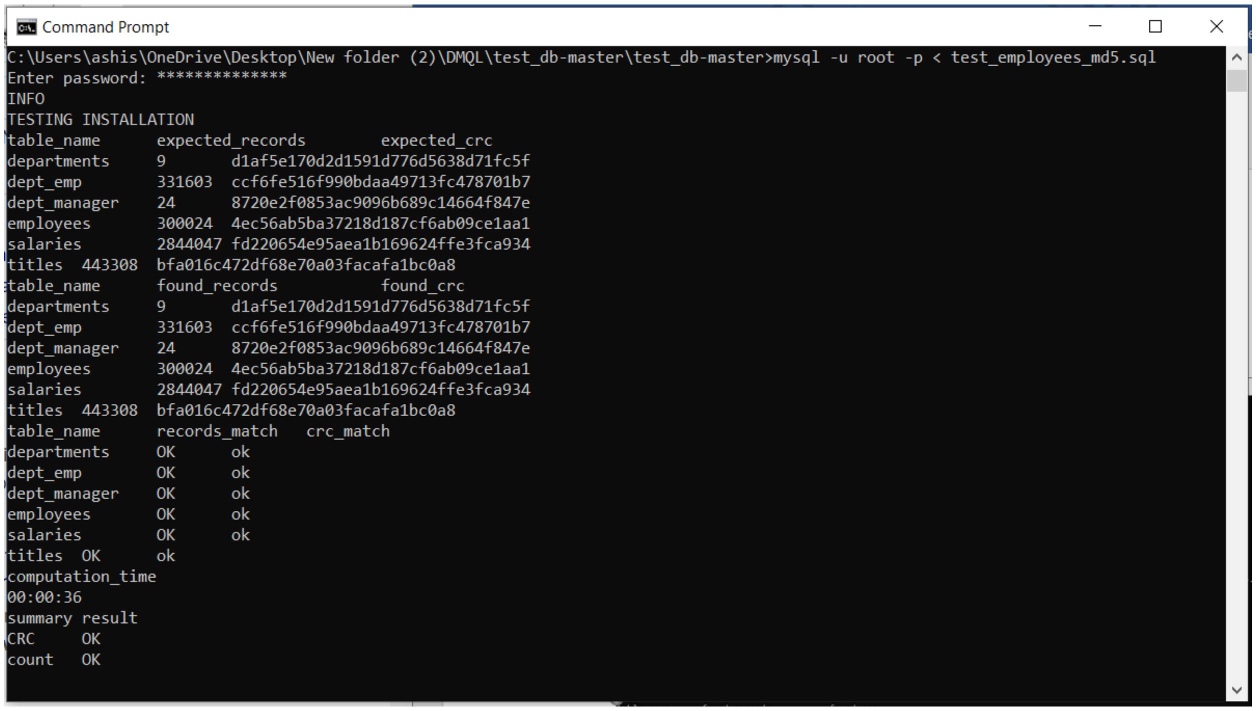
WE LOAD THE TITLES AS INFO AND ALSO THE DUMP FILE FROM THE LOAD_TITLES.

```
SELECT 'LOADING salaries' as 'INFO';

source load_salaries1.dump ;

source load_salaries2.dump ;

source load_salaries3.dump ;
```

WE LOAD THE SALARIES AS INFO AND ALSO THE DUMP FILE FROM THE SALARIES1, SALARIES2, SALARIES3.

```
source show_elapsed.sql ;
```

WE LOAD THE TIME ELAPSED.



```
C:\Users\ashis\OneDrive\Desktop\New folder (2)\DMQL\test_db-master\test_db-master>mysql -u root -p < test_employees_md5.sql
Enter password: **************
INFO
TESTING INSTALLATION
table_name      expected_records        expected_crc
departments     9       d1af5e170d2d1591d776d5638d71fc5f
dept_emp        331603  ccf6fe516f990bdaa49713fc478701b7
dept_manager    24      8720e2f0853ac9096b689c14664f847e
employees       300024  4ec56ab5ba37218d187cf6ab09ce1aa1
salaries        2844047 fd220654e95aea1b169624ffe3fca934
titles  443308  bfa016c472df68e70a03facafa1bc0a8
table_name      found_records           found_crc
departments     9       d1af5e170d2d1591d776d5638d71fc5f
dept_emp        331603  ccf6fe516f990bdaa49713fc478701b7
dept_manager    24      8720e2f0853ac9096b689c14664f847e
employees       300024  4ec56ab5ba37218d187cf6ab09ce1aa1
salaries        2844047 fd220654e95aea1b169624ffe3fca934
titles  443308  bfa016c472df68e70a03facafa1bc0a8
table_name      records_match   crc_match
departments     OK      ok
dept_emp        OK      ok
dept_manager    OK      ok
employees       OK      ok
salaries        OK      ok
titles  OK      ok
computation_time
00:00:36
summary result
CRC     OK
count   OK
```

IN STEP2: WE LOAD TEST_EMPLOYEES_MD5.SQL

USE employees;

WE CONNECT TO THE EMPLOYEES DATABASE AND USE IT.

SELECT 'TESTING INSTALLATION' as 'INFO';

RETREIVE INFORMATION FROM TESTING INSTALLATION AS INFORMATION.

DROP TABLE IF EXISTS expected_values, found_values;

WE DELETE THE TABLES (EXPECTED_VALUES, FOUND_VALUES) IF THEY ARE PRE-EXISTING WITH THE GIVEN NAME.

CREATE TABLE expected_values (

  table_name varchar(30) not null primary key,

  recs int not null,

  crc_sha varchar(100) not null,

  crc_md5 varchar(100) not null

);

WE CREATE EXPECTED_VALUES TABLE WITH THE FOLLOWING FIELDS:

TABLE_NAME WHICH IS OF TYPE VARCHAR AND DOES NOT ACCEPT NULL VALUES,

RECS WHICH IS OF TYPE INTEGER AND DOES NOT ACCEPT NULL VALUES,

CRC_SHA WHICH IS OF TYPE VARCHAR AND DOES NOT ACCEPT NULL VALUES,

CRC_MD5 WHICH IS OF TYPE VARCHAR AND DOES NOT ACCEPT NULL VALUES.

CREATE TABLE found_values LIKE expected_values;

WE CREATE FOUND_VALUES TABLE JUST LIKE THE EXPECTED_VALUES TABLE.

```
INSERT INTO `expected_values` VALUES

('employees',   300024,'4d4aa689914d8fd41db7e45c2168e7dcb9697359',

                 '4ec56ab5ba37218d187cf6ab09ce1aa1'),

('departments',      9,'4b315afa0e35ca6649df897b958345bcb3d2b764',

                 'd1af5e170d2d1591d776d5638d71fc5f'),

('dept_manager',    24,'9687a7d6f93ca8847388a42a6d8d93982a841c6c',

                 '8720e2f0853ac9096b689c14664f847e'),

('dept_emp',    331603, 'd95ab9fe07df0865f592574b3b33b9c741d9fd1b',

                 'ccf6fe516f990bdaa49713fc478701b7'),

('titles',      443308,'d12d5f746b88f07e69b9e36675b6067abb01b60e',

                 'bfa016c472df68e70a03facafa1bc0a8'),

('salaries',   2844047,'b5a1785c27d75e33a4173aaa22ccf41ebd7d4a9f',

                 'fd220654e95aea1b169624ffe3fca934');
```

WE NOW POPULATE THE EXPECTED_VALUES TABLE WITH THE TABLE_NAMES AND ITS CORRESPONDING VALUES.

```
SELECT table_name, recs AS expected_records, crc_md5 AS expected_crc FROM expected_values;
```

WE NOW RETREIVE INFORMATION ON THE TABLE_NAMES AND RECS AS EXPECTED_RECORDS

RETREIVE INFORMATION ON CRC_MD5 AS EXPECTED_CRC FROM THE TABLE EXPECTED VALUES.

```
DROP TABLE IF EXISTS tchecksum;
```

DELETE TABLE TCHECKSUM IF IT EXISTS

```
CREATE TABLE tchecksum (chk char(100));
```

WE CREATE TABLE TCHECKSUM

```
SET @crc= ' ';
```

WE SET THE CYCLIC REDUNDANCY (validation mechanism to detect corruption when a *page* in a *tablespace* is read from disk into the InnoDB *buffer pool*) CHECK VALUE AS EMPTY STRING.

```
INSERT INTO tchecksum

   SELECT @crc := MD5(CONCAT_WS('#',@crc,

            emp_no,birth_date,first_name,last_name,gender,hire_date))

   FROM employees ORDER BY emp_no;

INSERT INTO found_values VALUES ('employees', (SELECT COUNT(*) FROM employees), @crc,@crc);
```

WE NOW POPULATE THE TCHECKSUM TABLE WITH THE VALUES FROM EMPLOYEES TABLE AND INSERT INTO COLUMN FOUND VALUES THE COUNT OF THE EMPLOYEES FROM THE EMPLOYEES TABLE WHICH ARE ORDERED BY THE EMP_NO AND ALSO POPULATE THE CRC WITH THE MD5 HASHED COLUMN (WHICH CONTAINS THE STRING WHICH IS CONCATENATED VALUE OF CRC, EMP_NO, BIRTH_DATE, FIRST_NAME, LAST_NAME, GENDER, HIRE_DATE).

SET @crc = ' ';

WE SET THE CYCLIC REDUNDANCY (validation mechanism to detect corruption when a *page* in a *tablespace* is read from disk into the InnoDB *buffer pool*) CHECK VALUE AS EMPTY STRING.

INSERT INTO tchecksum

    SELECT @crc := MD5(CONCAT_WS('#',@crc, dept_no,dept_name))

    FROM departments ORDER BY dept_no;

INSERT INTO found_values values ('departments', (SELECT COUNT(*) FROM departments), @crc,@crc);

WE NOW POPULATE THE TCHECKSUM TABLE WITH THE VALUES FROM DEPARTMENT TABLE AND INSERT INTO COLUMN FOUND VALUES - THE COUNT OF THE DEPARTMENTS FROM THE DEPARTMENTS TABLE WHICH ARE ORDERED BY THE DEPT_NO AND ALSO POPULATE THE CRC WITH THE MD5 HASHED COLUMN (WHICH CONTAINS THE STRING WHICH IS CONCATENATED VALUE OF CRC, DEPT_NO, DEPT_NAME).

SET @crc = ' ';

WE SET THE CYCLIC REDUNDANCY (validation mechanism to detect corruption when a *page* in a *tablespace* is read from disk into the InnoDB *buffer pool*) CHECK VALUE AS EMPTY STRING.

INSERT INTO tchecksum

    SELECT @crc := MD5(CONCAT_WS('#',@crc, dept_no,emp_no, from_date,to_date))

    FROM dept_manager ORDER BY dept_no,emp_no;

INSERT INTO found_values values ('dept_manager', (SELECT COUNT(*) FROM dept_manager), @crc,@crc);

WE NOW POPULATE THE TCHECKSUM TABLE WITH THE VALUES FROM DEPARTMENT MANAGER TABLE AND INSERT INTO COLUMN FOUND VALUES THE COUNT OF THE DEPARTMENT_MANAGER FROM THE DEPARTMENT_MANAGER TABLE WHICH ARE ORDERED BY THE DEPT_NO AND EMPLOYEE_NO AND ALSO POPULATE THE CRC WITH THE MD5 HASHED COLUMN (WHICH CONTAINS THE STRING WHICH IS CONCATENATED VALUE OF CRC, DEPT_NO, EMP_NO, FROM_DATE, TO_DATE).

SET @crc = ' ';

WE SET THE CYCLIC REDUNDANCY (validation mechanism to detect corruption when a *page* in a *tablespace* is read from disk into the InnoDB *buffer pool*) CHECK VALUE AS EMPTY STRING.

INSERT INTO tchecksum

    SELECT @crc := MD5(CONCAT_WS('#',@crc, dept_no,emp_no, from_date,to_date))

    FROM dept_emp ORDER BY dept_no,emp_no;

INSERT INTO found_values values ('dept_emp', (SELECT COUNT(*) FROM dept_emp), @crc,@crc);

WE NOW POPULATE THE TCHECKSUM TABLE WITH THE VALUES FROM DEPARTMENT EMPLOYEE TABLE AND INSERT INTO COLUMN FOUND VALUES THE COUNT OF THE DEPARTMENTS FROM THE DEPARTMENT MANAGER TABLE WHICH ARE ORDERED BY THE DEPT_NO AND EMPLOYEE_NO AND ALSO POPULATE THE CRC WITH THE MD5 HASHED COLUMN (WHICH CONTAINS THE STRING WHICH IS CONCATENATED VALUE OF CRC, DEPT_NO, EMP_NO, FROM_DATE, TO_DATE).

SET @crc = ' ';

WE SET THE CYCLIC REDUNDANCY (validation mechanism to detect corruption when a *page* in a *tablespace* is read from disk into the InnoDB *buffer pool*) CHECK VALUE AS EMPTY STRING.

INSERT INTO tchecksum

    SELECT @crc := MD5(CONCAT_WS('#',@crc, emp_no, title, from_date,to_date))

```
    FROM titles order by emp_no,title,from_date;

INSERT INTO found_values values ('titles', (SELECT COUNT(*) FROM titles), @crc,@crc);
```

WE NOW POPULATE THE TCHECKSUM TABLE WITH THE VALUES FROM TITLES TABLE AND INSERT INTO COLUMN FOUND VALUES THE COUNT OF THE TITLES FROM THE TITLES TABLE WHICH ARE ORDERED BY THE EMPLOYEE_NO, TITLE AND FROM_DATE AND ALSO POPULATE THE CRC WITH THE MD5 HASHED COLUMN (WHICH CONTAINS THE STRING WHICH IS CONCATENATED VALUE OF CRC, EMP_NO, TITLE, FROM_DATE, TO_DATE).

```
SET @crc = ' ';
```

WE SET THE CYCLIC REDUNDANCY (validation mechanism to detect corruption when a *page* in a *tablespace* is read from disk into the InnoDB *buffer pool*) CHECK VALUE AS EMPTY STRING.

```
INSERT INTO tchecksum

    SELECT @crc := MD5(CONCAT_WS('#',@crc, emp_no, salary, from_date,to_date))

    FROM salaries order by emp_no,from_date,to_date;

INSERT INTO found_values values ('salaries', (SELECT COUNT(*) FROM salaries), @crc,@crc);
```

WE NOW POPULATE THE TCHECKSUM TABLE WITH THE VALUES FROM SALARIES TABLE AND INSERT INTO COLUMN FOUND VALUES THE COUNT OF THE TITLES FROM THE TITLES TABLE WHICH ARE ORDERED BY THE EMPLOYEE_NO, FROM_DATE, TO_DATE AND ALSO POPULATE THE CRC WITH THE MD5 HASHED COLUMN (WHICH CONTAINS THE STRING WHICH IS CONCATENATED VALUE OF CRC, EMP_NO, SALARY, FROM_DATE, TO_DATE).

```
DROP TABLE tchecksum;
```

DROP TABLE TCHECKSUM

```
SELECT table_name, recs as 'found_records   ', crc_md5 as found_crc from found_values;
```

WE NOW CREATE A VIRTUAL TABLE WITH (TABLE_NAME AND RECORDS) AS FOUND_RECORDS, (CRC_MD5) AS FOUND_CRC FROM FOUND_VALUES.

```
SELECT

    e.table_name,

    IF(e.recs=f.recs,'OK', 'not ok') AS records_match,

    IF(e.crc_md5=f.crc_md5,'ok','not ok') AS crc_match

from

    expected_values e INNER JOIN found_values f USING (table_name);
```

WE NOW CREATE A VIRTUAL TABLE WITH TABLE_NAME WHICH CONTAINS THE INNER JOIN OF EXPECTED_VALUES AND FOUND_VALUES WHOSE RECORDS AND CRC MATCHES EACH OTHER.

```
set @crc_fail=(select count(*) from expected_values e inner join found_values f on (e.table_name=f.table_name) where
f.crc_md5 != e.crc_md5);
```

WE NOW SET CRC TO FAIL WHEN THE INNER JOIN OF (EXPECTED_VALUES AND FOUND_VALUES) OF CRC DOES NOT MATCH EACH OTHER. THE CRC_FAIL COUNTS THE NUMBER OF OCCURANCES OF CRC FAILS.

```
set @count_fail=(select count(*) from expected_values e inner join found_values f on (e.table_name=f.table_name) where
f.recs != e.recs);
```

WE NOW SET COUNT TO FAIL WHEN THE INNER JOIN OF (EXPECTED_VALUES AND FOUND_VALUES) OF RECORDS DOES NOT MATCH EACH OTHER. THE COUNT_FAIL COUNTS THE NUMBER OF OCCURANCES OF COUNT FAILS.

```
select timediff(
    now(),
    (select create_time from information_schema.tables where table_schema='employees' and table_name='expected_values')
) as computation_time;
```

DISPLAY THE TIME DIFFERENCE BETWEEN THE TIME OF CREATION OF THE EXPECTED VALUES COMPUTATION TO NOW AS THE COMPUTATION TIME.

```
DROP TABLE expected_values,found_values;
```

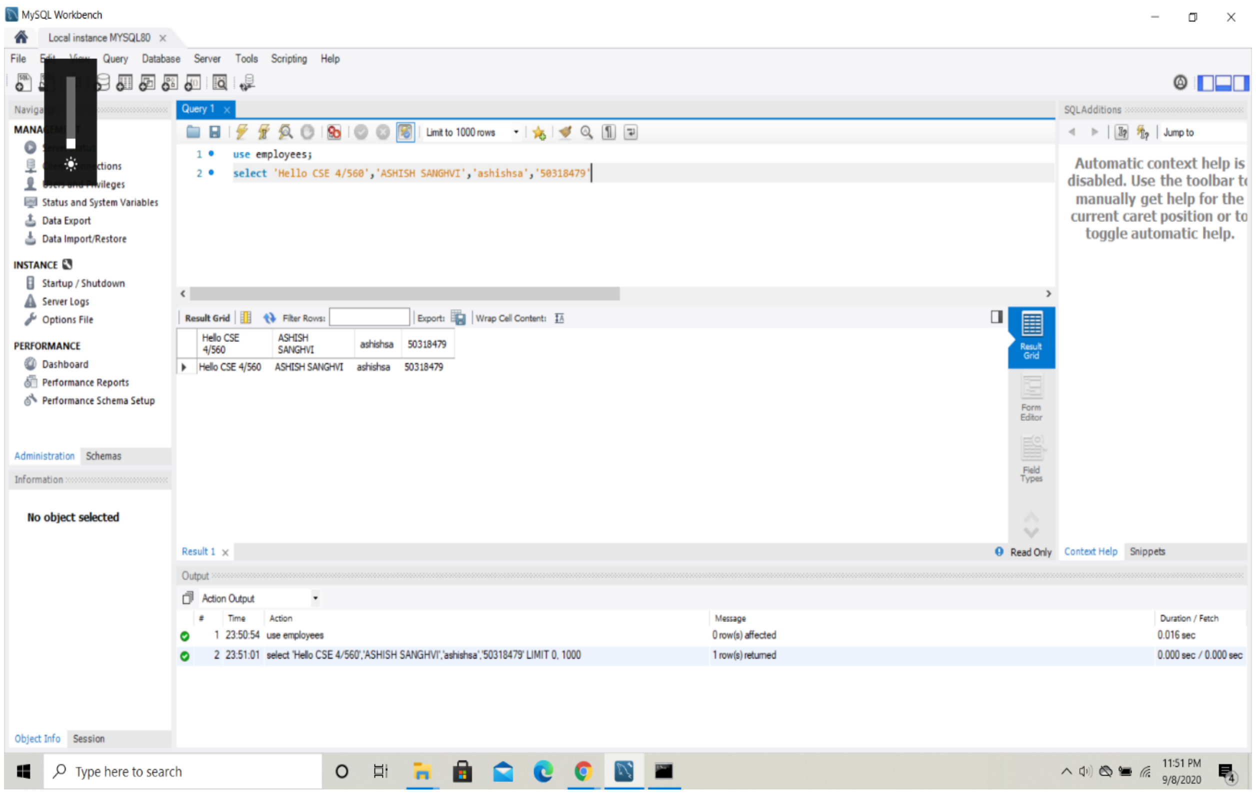
WE NOW DROP THE EXPECTED VALUE AND FOUND VALUE TABLES.

```
select 'CRC' as summary,  if(@crc_fail = 0, "OK", "FAIL" ) as 'result'

union all

select 'count', if(@count_fail = 0, "OK", "FAIL" );
```

IF WE GET THE SUM OF CRC_FAILS=0 DISPLAY OK, IF NOT DISPLAY FAIL.

IF WE GET THE SUM OF COUNT_FAILS=0 DISPLAY OK, IF NOT DISPLAY FAIL.

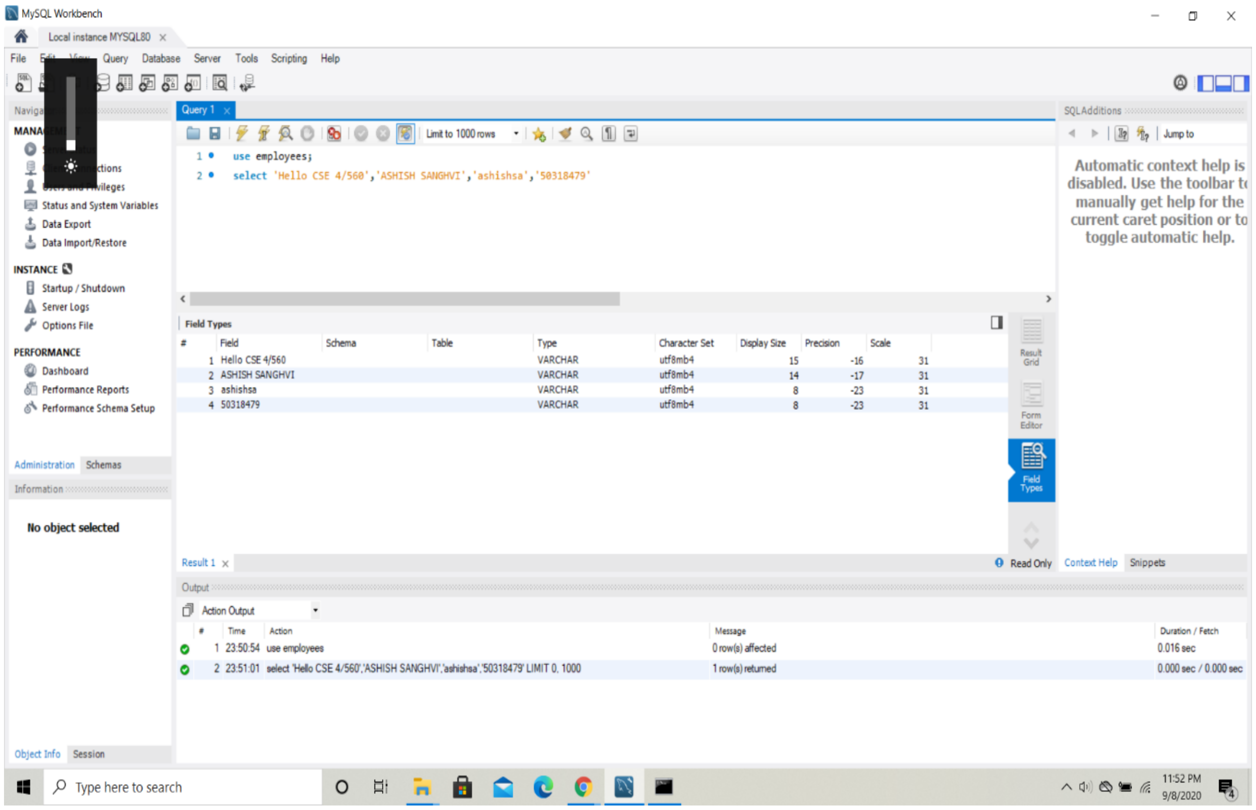
AND THEN UNION THE RESULTS AND PRINT IT.



```
use employees;
select 'Hello CSE 4/560','ASHISH SANGHVI','ashishsa','50318479'
```

```
Hello CSE 4/560   ASHISH SANGHVI   ashishsa   50318479
Hello CSE 4/560   ASHISH SANGHVI   ashishsa   50318479
```

```
#  Time      Action                                                             Message              Duration / Fetch
1  23:50:54  use employees                                                      0 row(s) affected    0.016 sec
2  23:51:01  select 'Hello CSE 4/560','ASHISH SANGHVI','ashishsa','50318479' LIMIT 0, 1000   1 row(s) returned    0.000 sec / 0.000 sec
```



```
use employees;
select 'Hello CSE 4/560','ASHISH SANGHVI','ashishsa','50318479'
```

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

Field Types

#	Field	Schema	Table	Type	Character Set	Display Size	Precision	Scale
1	Hello CSE 4/560			VARCHAR	utf8mb4	15	-16	31
2	ASHISH SANGHVI			VARCHAR	utf8mb4	14	-17	31
3	ashishsa			VARCHAR	utf8mb4	8	-23	31
4	50318479			VARCHAR	utf8mb4	8	-23	31

#	Time	Action	Message	Duration / Fetch
1	23:50:54	use employees	0 row(s) affected	0.016 sec
2	23:51:01	select 'Hello CSE 4/560','ASHISH SANGHVI','ashishsa','50318479' LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec

IN STEP3: WE RUN THE SQL CLIENT TO DISPLAY THE FOLLOWING INFORMATION

use employees;

WE USE EMPLOYEES DATABASE.

SELECT ’Hello CSE 4/560’, ’Your Name’, ’Your UBIT’, ’Person #’

WE RETREIVE ‘HELLO CSE 4/560’, ‘ASHISH SANGHVI’, ‘ashishsa’,‘50318479’.