**Name: - Akhand Pratap Singh**

**Sap ID: - 500085043**

**EXPERIMENT-7**

Title: To understand the concepts of Index.

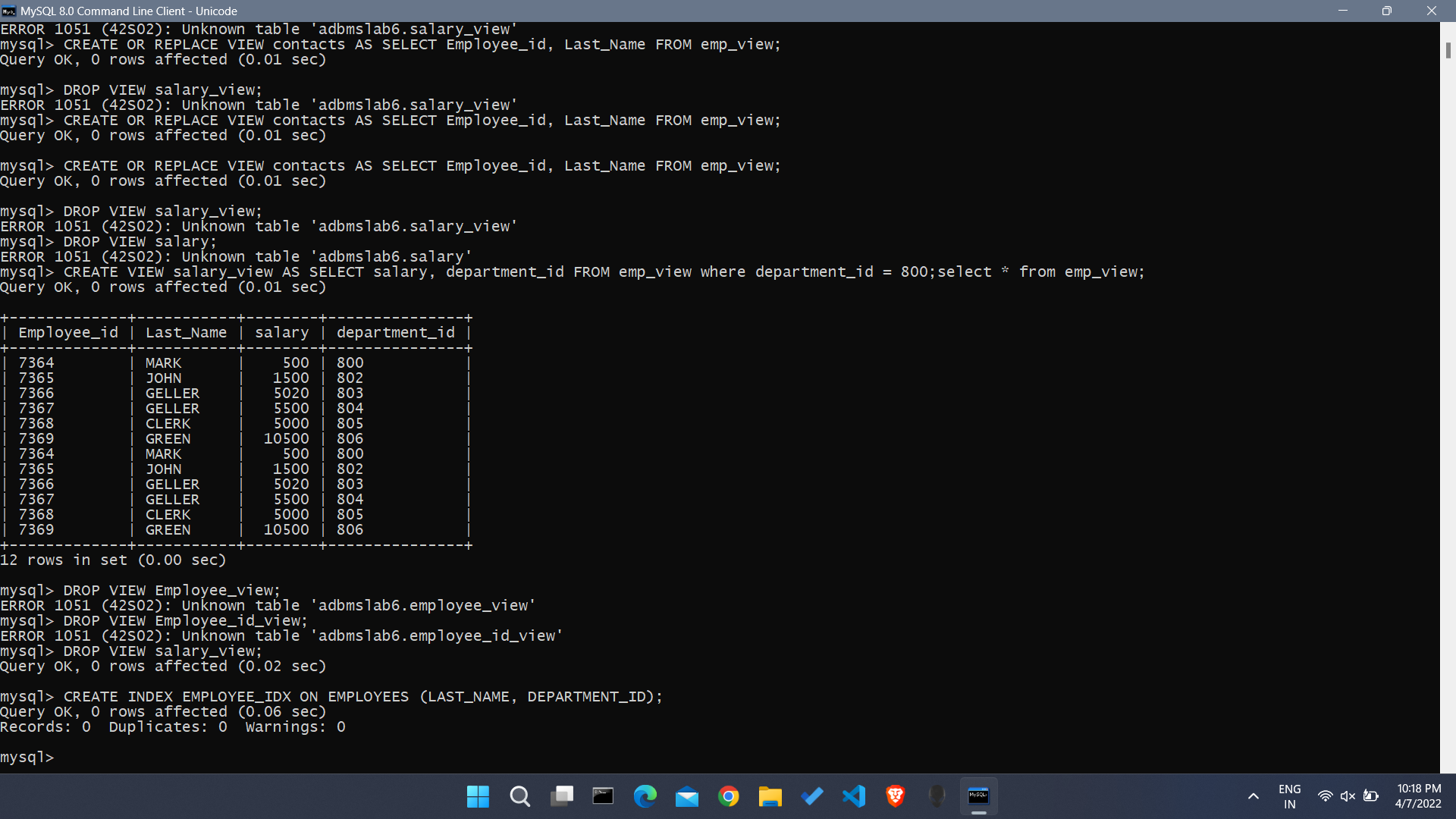
Q1. Create an index of name EMPLOYEE\_IDX on EMPLOYEES with column LAST\_NAME, DEPARTMENT\_ID.

Code:

--1

CREATE INDEX EMPLOYEE\_IDX ON EMPLOYEES (LAST\_NAME, DEPARTMENT\_ID);

Screenshot:



Q2. Find the ROWID for the above table and create a unique index on EMPLOYEE\_ID column of the EMPLOYEES.

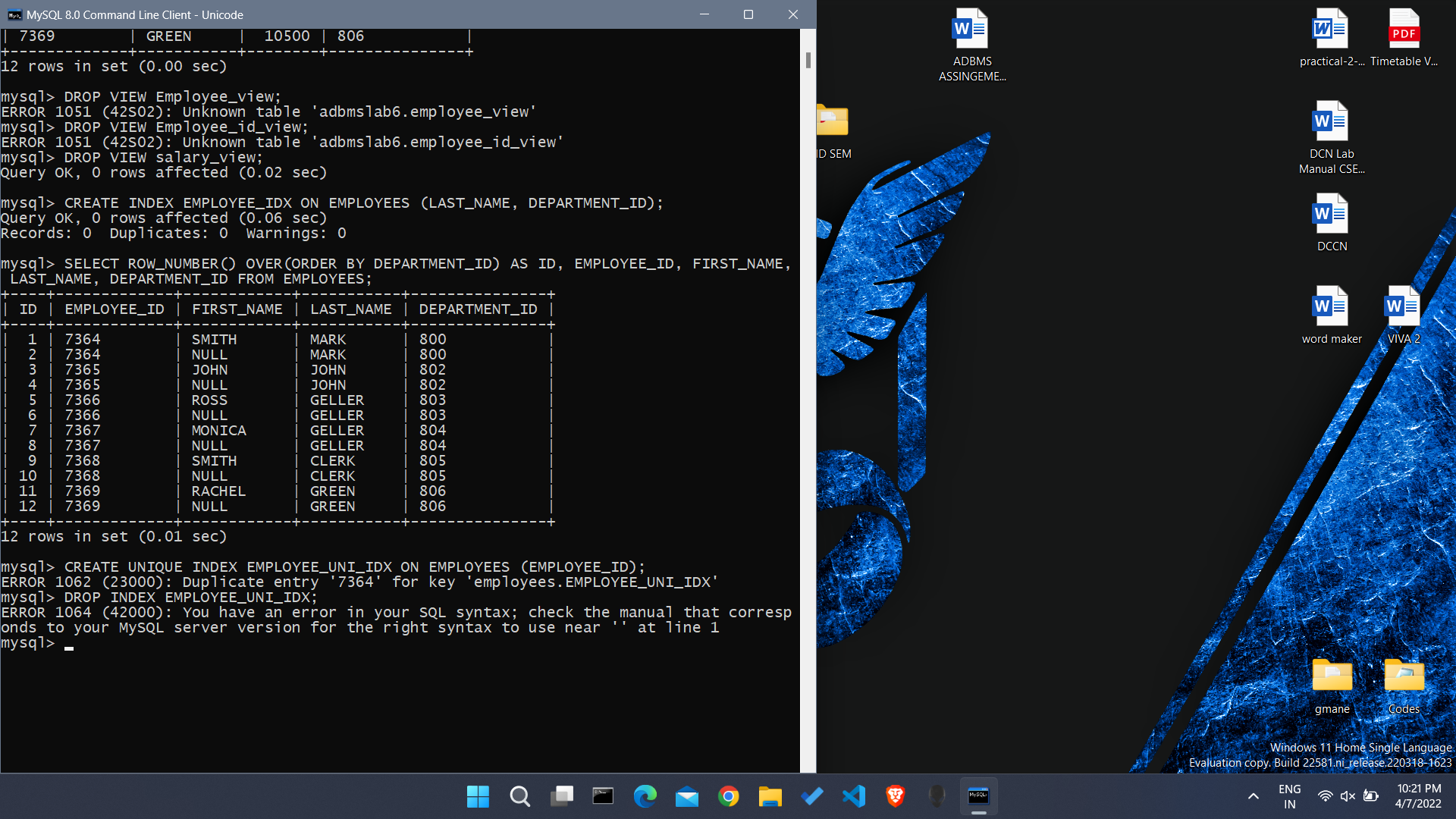
Code:

--2

SELECT ROW\_NUMBER() OVER(ORDER BY DEPARTMENT\_ID) AS ID, EMPLOYEE\_ID, FIRST\_NAME, LAST\_NAME, DEPARTMENT\_ID FROM EMPLOYEES;

CREATE UNIQUE INDEX EMPLOYEE\_UNI\_IDX ON EMPLOYEES (EMPLOYEE\_ID);

Screenshot:



Q3. Create a reverse index on EMPLOYEE\_ID column of the EMPLOYEES.

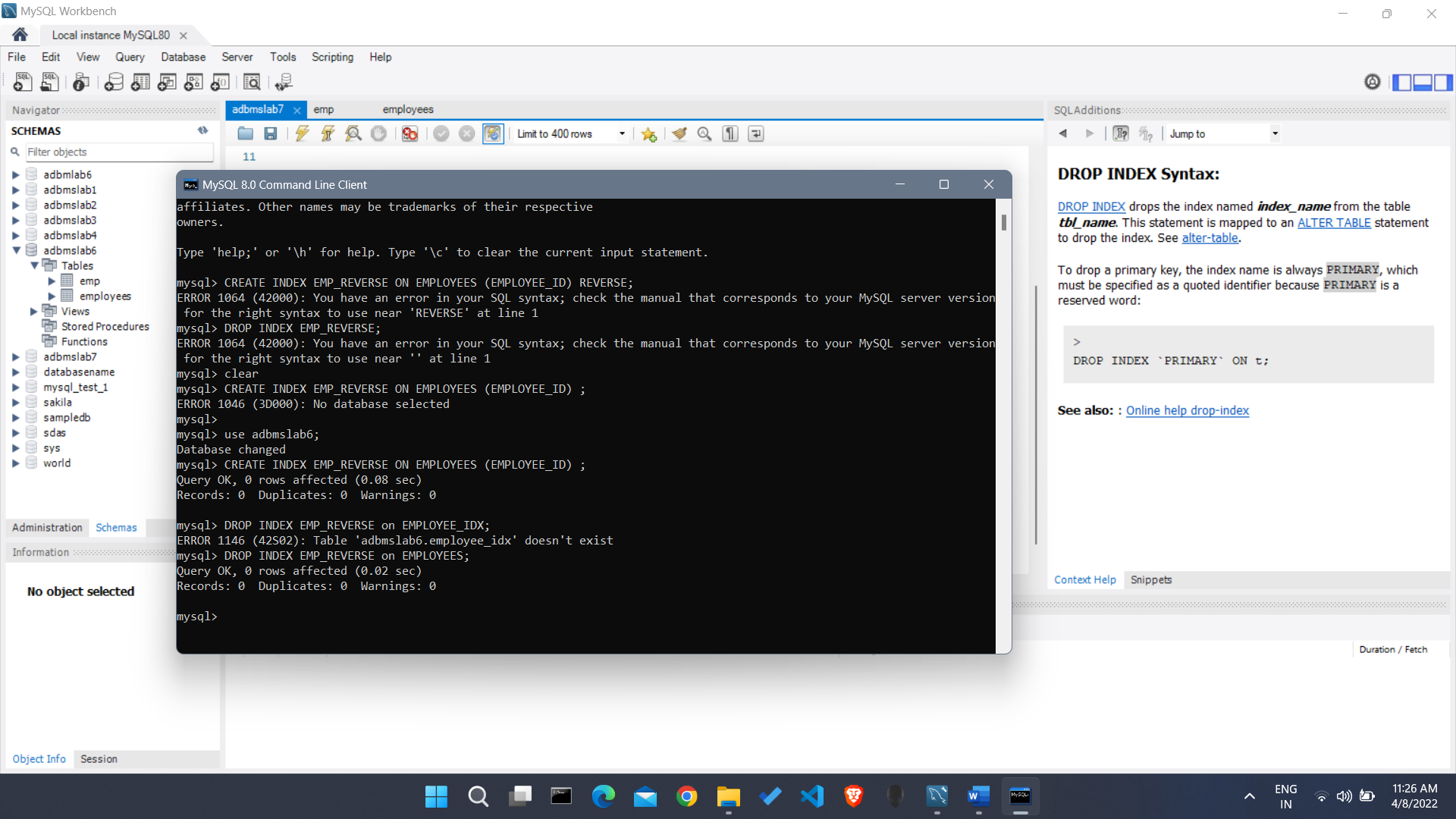
Code:

--3

CREATE INDEX EMP\_REVERSE ON EMPLOYEES (EMPLOYEE\_ID) REVERSE;

DROP INDEX EMP\_REVERSE;

Screenshot:



Q4. Create a unique and composite index on EMPLOYEE\_ID and check whether there is duplicity of tuples or not.

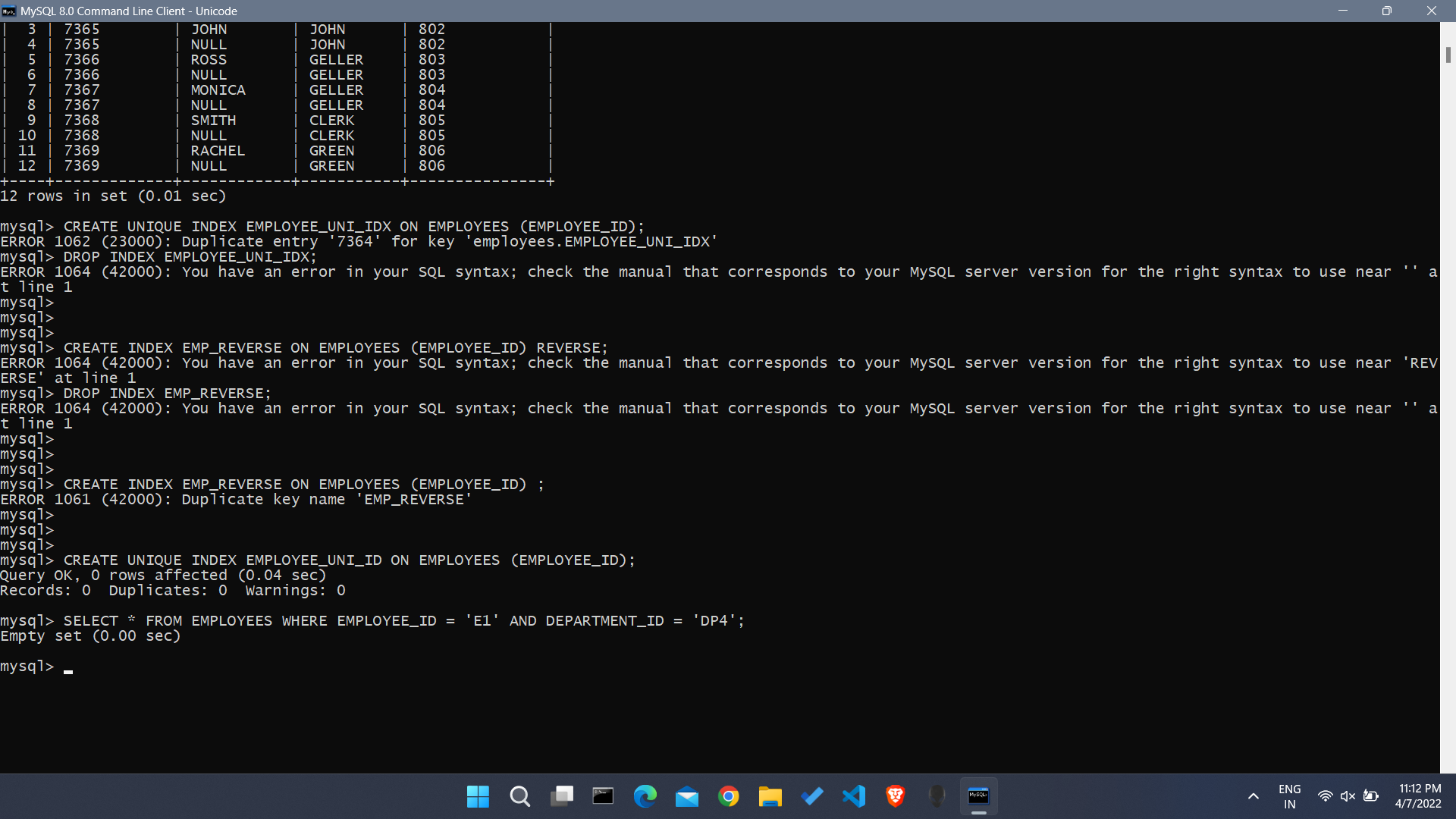
Code:

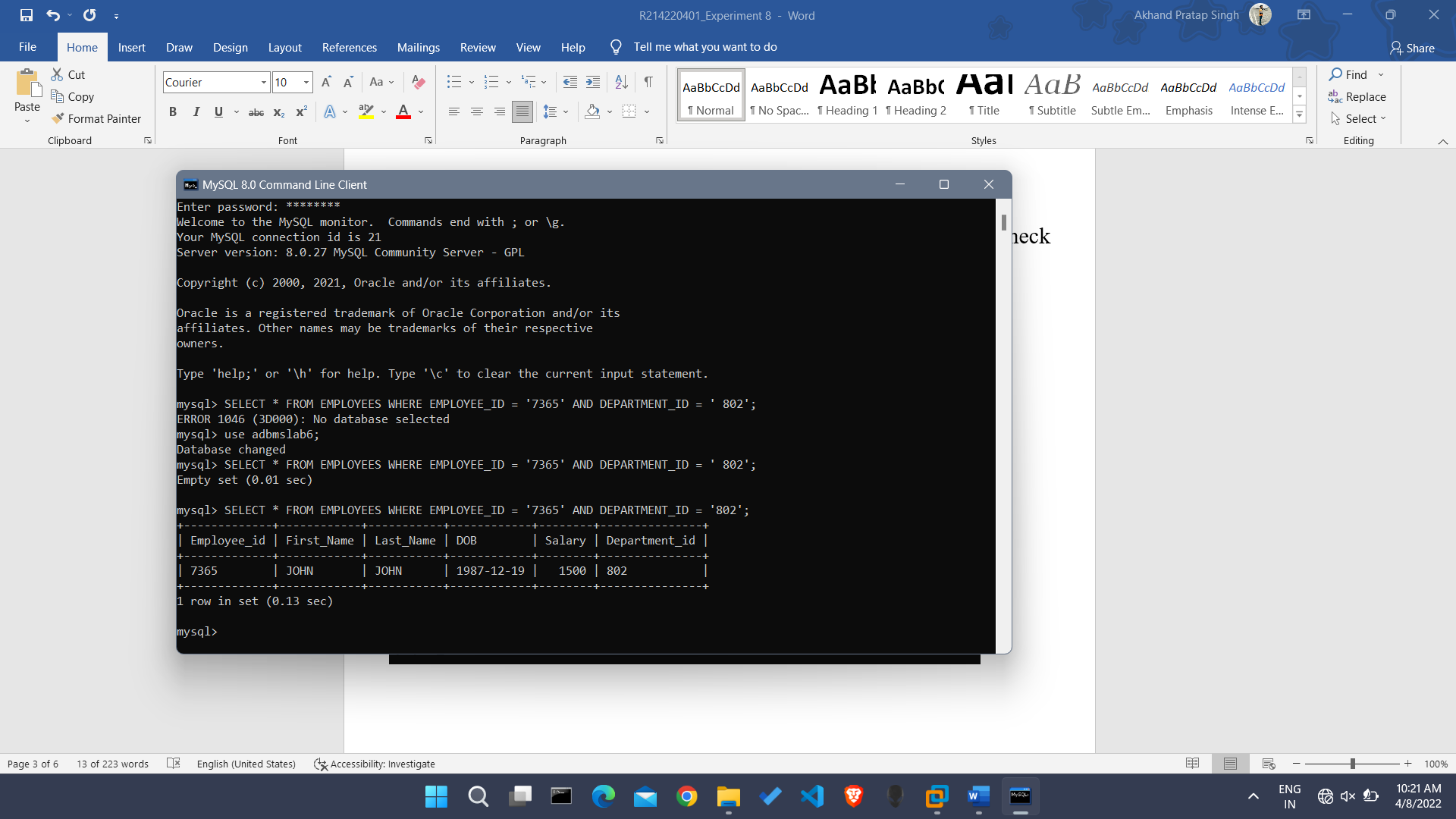
--4

CREATE UNIQUE INDEX EMPLOYEE\_UNI\_COMP ON EMPLOYEES (EMPLOYEE\_ID, DEPARTMENT\_ID);

SELECT \* FROM EMPLOYEES WHERE EMPLOYEE\_ID = '7365' AND DEPARTMENT\_ID = ' 802';

Screenshot:





Q5. Create Function-based indexes defined on the SQL functions UPPER(column\_name) or LOWER(column\_name) to facilitate case-insensitive searches on column LAST\_NAME.

Code:

--5

CREATE INDEX LAST\_NAME\_IDX ON EMPLOYEES (UPPER(LAST\_NAME));

CREATE INDEX LAST\_NAME\_IDX2 ON EMPLOYEES (lower (LAST\_NAME));

Screenshot:

Q6. Drop the function-based index on column LAST\_NAME.

Code:

--6

DROP INDEX LAST\_NAME\_IDX;

DROP INDEX LAST\_NAME\_IDX2;

Screenshot: