z

# **ENTERPRISE ORGANIZATION SYSTEMS MANAGEMENT**

Jagadeesh Kumar Periketi

L30063639

[jagadeeshkumarperi@lewisu.edu](mailto:jagadeeshkumarperi@lewisu.edu)

Rama Jaswanth Konakalla

L30062330

[ramajaswanthkonaka@lewisu.edu](mailto:ramajaswanthkonaka@lewisu.edu)

Shiva Chembeti

L30064354

[shivachembeti@lewisu.edu](mailto:shivachembeti@lewisu.edu)

# Table of Contents

1. Initial Proposal ------------------------------------------------------------3
2. Data Sources---------------------------------------------------------------3
3. Data Storage Alternatives------------------------------------------------3
4. Relational Database Design Process------------------------------------4
5. Relational Database Design----------------------------------------------7
6. Data Definition Language------------------------------------------------7
7. Data Manipulation Language--------------------------------------------10
8. Indexes----------------------------------------------------------------------17
9. Views------------------------------------------------------------------------18
10. Triggers----------------------------------------------------------------------19
11. Transactions-----------------------------------------------------------------19
12. Security----------------------------------------------------------------------20
13. Locking----------------------------------------------------------------------22
14. Backup-----------------------------------------------------------------------24
15. Python Programming------------------------------------------------------26
16. PHP/Java Programming---------------------------------------------------28
17. Suggested Future Work---------------------------------------------------34
18. Activity Log-----------------------------------------------------------------35

## Initial Proposal:

Here we want to store different types of data like images, text, and XML, and JSON data according to the idea because we need images to store employee details, our main interest is to store employee details and provide a good service as an enterprise organization, it is important to store to maintain a crystal clear organization in the industry because, if we take an example in big companies like Microsoft, meta and apple this type companies follow certain rules for their employee from prohibition period to full-time employee,

the complete data comes from different types so here we are using two types of data one is no-sql and another is traditional way SQL, and we want to develop this as web application with good user interface.

## Data Sources

Here we are using two data sources one is firebase to maintain employees id, email, and password, and we are generating tokens, we are using a firebase data source to secure data and other details we are using SQL as main data source.

## Data Storage Alternatives

A non-relational database stores data in a non-tabular form, and a relational database stores data from traditional SQL-based relational database structures. It does not follow the relational model provided by traditional relational database management systems else we can migrate into different relational databases like AWS, some of the advantages are we can execute queries fast for example if we have millions of data it’s hard to get output in normal ways and it’s a time complexity and also its very expensive.

We have different types of alternatives to build relational databases to store the data. We have such type of database is called NoSQL Database. One database is called Mongo DB.

Since in MongoDB we save using JSON structure, our json files can be straightaway stored in the mongo db.

Mongo DB advantages are easy to handle data which has large volumes with high speed, this database will store unstructured data, structured data and semi-structured data. This is a developer friendly to write queries easily and this will take the full advantage of the cloud with having zero downtime.

MongoDB is faster than relational databases due to its ability to handle large amounts of unstructured data when it comes to speed. It uses slave replication, master replication to process vast amounts of unstructured data and offers the freedom to use multiple data types that are better than the rigidity of relational databases.

* In Memory Databases: In-memory databases are faster than traditional databases because they require fewer CPU instructions. They also eliminate the time it takes to access data from a disk. In-memory databases are more volatile than traditional databases because data is lost when there is a loss of power or the computer's RAM crashes.

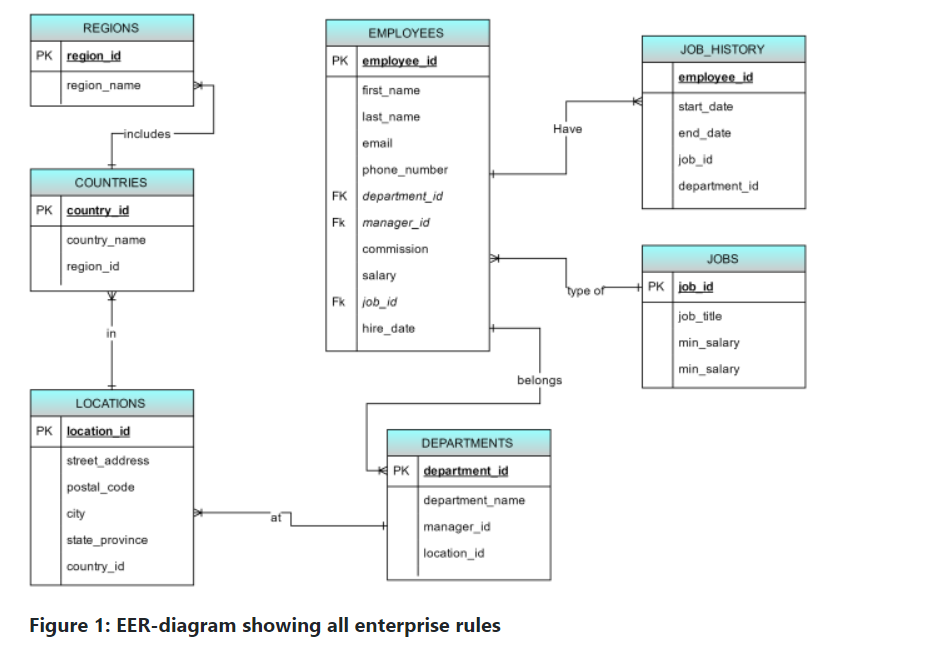
These can be used for providing faster response to the end users, but not as main data storage or warehouse.

Data storage in an in-memory database relies on a computer's random-access memory (RAM) or main memory instead of traditional disk drives. Data is loaded into an in-memory database in a compressed and non-relational format. The data is in a directly usable format without the barrier of compression or encryption.

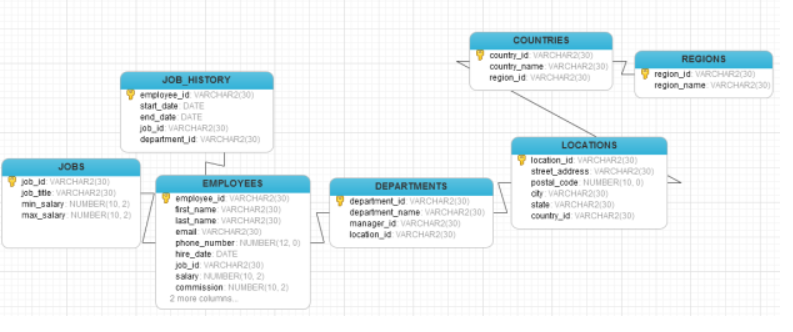
## Relational Database Design Process:



Below is the conceptual design for the enterprise organization system database. This has been built in the online tool which portrays the actual design of the organization structure.



## Relational Database Design:



## Data Definition Language (DDL) Scripts:

In structured query language or else we call it as database language, here we have some operations to perform on the database which are divided into multiple types based on their command behavior. DDL is a set of SQL commands used to create, modify, and delete database structures but not data. These commands are normally not used by a general user, who should be accessing the database via an application.

List of DDL commands:

* [CREATE](https://www.geeksforgeeks.org/sql-create/): Command used to create the db or its objects.
* [DROP](https://www.geeksforgeeks.org/sql-drop-truncate/): Command used to delete objects from the database.
* [ALTER](https://www.geeksforgeeks.org/sql-alter-add-drop-modify/): Command used to alter the structure of the database.
* [TRUNCATE](https://www.geeksforgeeks.org/sql-drop-truncate/): Command used to remove all records from a table, which includes all spaces allocated are removed.
* [COMMENT](https://www.geeksforgeeks.org/sql-comments/): This is used to add comments to the data dictionary.
* [RENAME](https://www.geeksforgeeks.org/sql-alter-rename/): This is used to rename an object existing in the database.

Table structure for COUNTRIES

DROP TABLE “MYDB”.”COUNTRIES”;

CREATE TABLE “MYDB”.”COUNTRIES” (

“country\_id” VARCHAR2(30) NOT NULL ,

“country\_name” VARCHAR2(30) NULL ,

“region\_id” VARCHAR2(30) NULL

)

Table structure for DEPARTMENTS

DROP TABLE “MYDB”.”DEPARTMENTS”;

CREATE TABLE “MYDB”.”DEPARTMENTS” (

“department\_id” VARCHAR2(30) NOT NULL ,

“department\_name” VARCHAR2(30) NULL ,

“manager\_id” VARCHAR2(30) NULL ,

“location\_id” VARCHAR2(30) NULL

)

Table structure for EMPLOYEES

DROP TABLE “MYDB”.”EMPLOYEES”;

CREATE TABLE “MYDB”.”EMPLOYEES” (

“employee\_id” VARCHAR2(30) NOT NULL ,

“first\_name” VARCHAR2(30) NULL ,

“last\_name” VARCHAR2(30) NULL ,

“email” VARCHAR2(30) NULL ,

“phone\_number” NUMBER(12) NULL ,

“hire\_date” DATE NULL ,

“job\_id” VARCHAR2(30 BYTE) NULL ,

“salary” NUMBER(10,2) NULL ,

“commission” NUMBER(10,2) NULL ,

“manager\_id” VARCHAR2(30) NULL ,

“department\_id” VARCHAR2(30) NULL

)

Table structure for JOB\_HISTORY

DROP TABLE “MYDB”.”JOB\_HISTORY”;

CREATE TABLE “MYDB”.”JOB\_HISTORY” (

“employee\_id” VARCHAR2(30) NOT NULL ,

“start\_date” DATE NULL ,

“end\_date” DATE NULL ,

“job\_id” VARCHAR2(30) NULL ,

“department\_id” VARCHAR2(30) NOT NULL

)

Table structure for JOBS

DROP TABLE “MYDB”.”JOBS”;

CREATE TABLE “MYDB”.”JOBS” (

“job\_id” VARCHAR2(30) NOT NULL ,

“job\_title” VARCHAR2(30) NULL ,

“min\_salary” NUMBER(10,2) NULL ,

“max\_salary” NUMBER(10,2) NULL

)

Table structure for LOCATIONS

DROP TABLE “MYDB”.”LOCATIONS”;

CREATE TABLE “MYDB”.”LOCATIONS” (

“location\_id” VARCHAR2(30) NOT NULL ,

“street\_address” VARCHAR2(30) NULL ,

“postal\_code” NUMBER(10) NULL ,

“city” VARCHAR2(30) NULL ,

“state” VARCHAR2(30) NULL ,

“country\_id” VARCHAR2(30) NULL

)

Table structure for REGIONS

DROP TABLE “MYDB”.”REGIONS”;

CREATE TABLE “MYDB”.”REGIONS” (

“region\_id” VARCHAR2(30) NOT NULL ,

“region\_name” VARCHAR2(30) NULL

)

## Data Manipulation Language Scripts:

The SQL commands that checks with the manipulation or modification of records present in the db relates to DML or Data Manipulation Language which includes mostly the SQL statements. This is the component of SQL statement that controls access to data and to the database.

List of DML commands:

* [INSERT](https://www.geeksforgeeks.org/sql-insert-statement/) : It is used to insert data into a table.
* [UPDATE](https://www.geeksforgeeks.org/sql-update-statement/): It is used to update existing data within a table.
* [DELETE](https://www.geeksforgeeks.org/sql-delete-statement/) : It is used to delete records from a database table.
* [LOCK:](https://www.geeksforgeeks.org/sql-lock-table/) Table control concurrency.

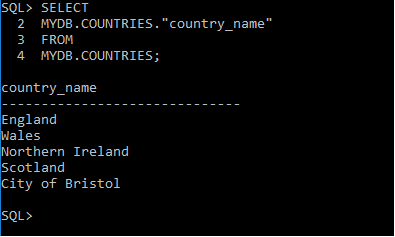
**Query 1**

SELECT

MYDB.COUNTRIES.”country\_name”

FROM

MYDB.COUNTRIES



**Query 2**

SELECT

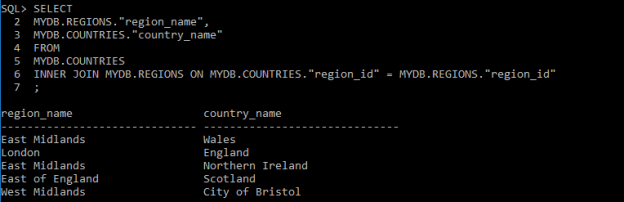
MYDB.REGIONS.”region\_name”,

MYDB.COUNTRIES.”country\_name”

FROM

MYDB.COUNTRIES

INNER JOIN MYDB.REGIONS ON MYDB.COUNTRIES.”region\_id” = MYDB.REGIONS.”region\_id”



**Query 3**

SELECT

MYDB.JOB\_HISTORY.”start\_date”,

MYDB.JOB\_HISTORY.”end\_date”,

MYDB.EMPLOYEES.”first\_name”,

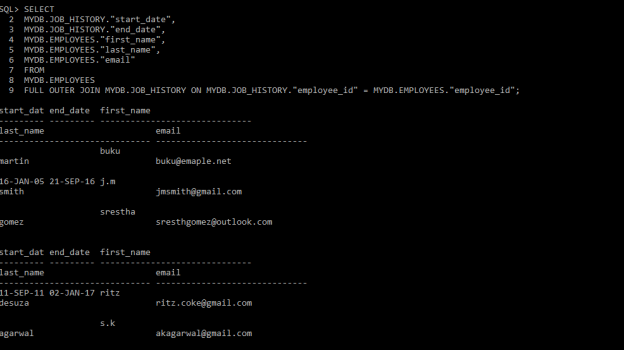
MYDB.EMPLOYEES.”last\_name”,

MYDB.EMPLOYEES.”email”

FROM

MYDB.EMPLOYEES

FULL OUTER JOIN MYDB.JOB\_HISTORY ON MYDB.JOB\_HISTORY.”employee\_id” = MYDB.EMPLOYEES.”employee\_id”



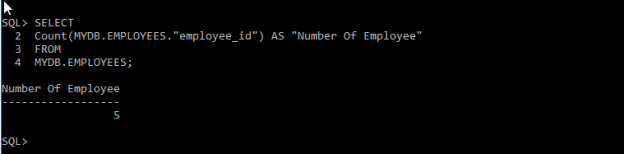
**Query 4**

SELECT

Count(MYDB.EMPLOYEES.”employee\_id”) AS “Number Of Employee”

FROM

MYDB.EMPLOYEES



**Query 5**

SELECT

MYDB.EMPLOYEES.”first\_name”,

MYDB.EMPLOYEES.”last\_name”,

MYDB.EMPLOYEES.”email”,

MYDB.EMPLOYEES.”phone\_number”,

MYDB.EMPLOYEES.”hire\_date”,

MYDB.EMPLOYEES.”salary”,

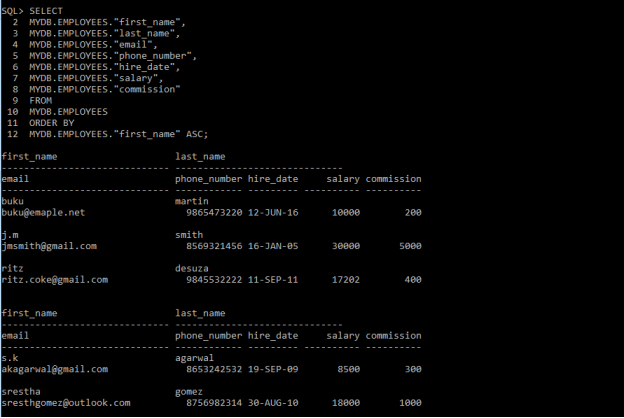
MYDB.EMPLOYEES.”commission”

FROM

MYDB.EMPLOYEES

ORDER BY

MYDB.EMPLOYEES.”first\_name” ASC



**Query 6**

SELECT

MYDB.EMPLOYEES.”first\_name”,

MYDB.EMPLOYEES.”last\_name”,

MYDB.EMPLOYEES.”email”,

MYDB.EMPLOYEES.”phone\_number”,

MYDB.EMPLOYEES.”hire\_date”,

MYDB.EMPLOYEES.”salary”,

MYDB.EMPLOYEES.”commission”

FROM

MYDB.EMPLOYEES

WHERE

MYDB.EMPLOYEES.”email” LIKE ‘%gmail%’



**Query 7**

SELECT

MYDB.EMPLOYEES.”first\_name”,

MYDB.EMPLOYEES.”last\_name”,

MYDB.EMPLOYEES.”email”,

MYDB.EMPLOYEES.”phone\_number”

FROM

MYDB.EMPLOYEES

INNER JOIN MYDB.JOB\_HISTORY ON MYDB.JOB\_HISTORY.”employee\_id” = MYDB.EMPLOYEES.”employee\_id”

WHERE

MYDB.JOB\_HISTORY.”employee\_id” IN (MYDB.EMPLOYEES.”employee\_id”)



**Query 8**

MYDB.EMPLOYEES.”email”,

MYDB.EMPLOYEES.”phone\_number”,

MYDB.EMPLOYEES.”hire\_date”,

MYDB.EMPLOYEES.”job\_id”,

MYDB.EMPLOYEES.”salary”,

MYDB.EMPLOYEES.”commission”,

MYDB.EMPLOYEES.”manager\_id”,

MYDB.EMPLOYEES.”department\_id”,

MYDB.EMPLOYEES.”employee\_id”

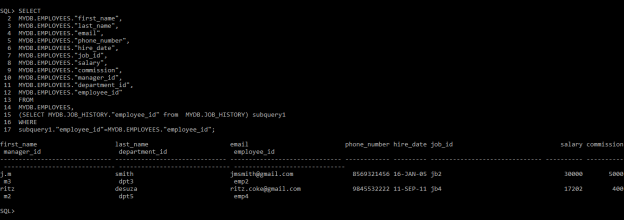
FROM

MYDB.EMPLOYEES,

(SELECT MYDB.JOB\_HISTORY.”employee\_id” fromÂ  MYDB.JOB\_HISTORY) subquery1

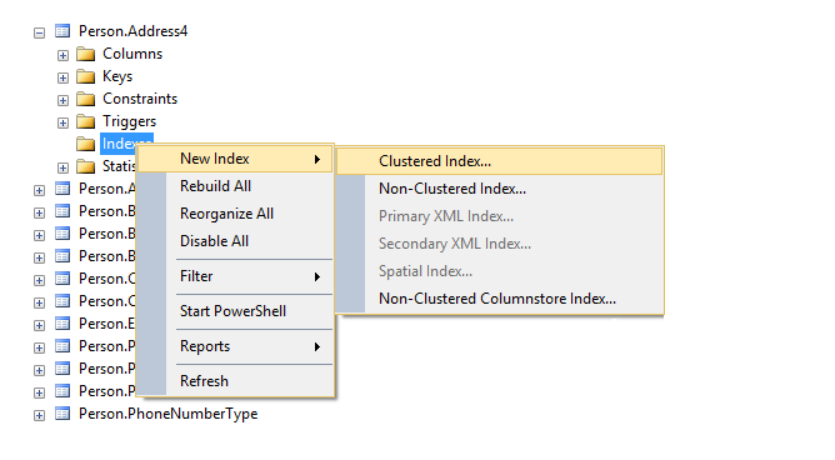
WHERE

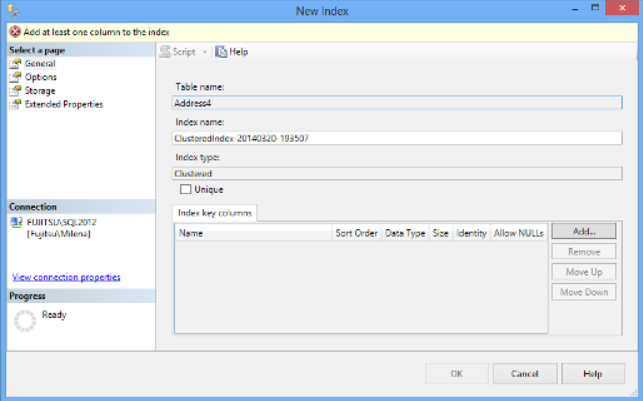
subquery1.”employee\_id”=MYDB.EMPLOYEES.”employee\_id”

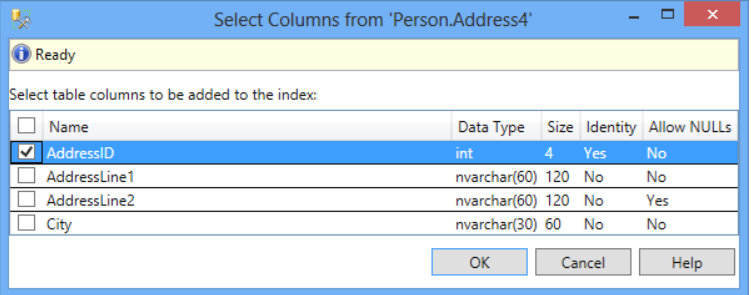


Indexes:

Generally, indexes are most commonly used concept in the IT organizations where large data is being processed, by creating index on said columns the query execution time decreases and performance increases. These indexes will get created automatically whenever a primary key and unique constraints are created on a table. Non clustered indexes are created at the time of table creation and when you create an primary key or a unique constraint immediately the clustered index will get created.







Views:

We can create View using **CREATE VIEW** statement. A View can be created from a single table or multiple tables.

CREATE VIEW view\_name AS

SELECT column1, column2.....

FROM table\_name

WHERE condition;

**view\_name**: Name for the View

**table\_name**: Name of the table

**condition**: Condition to select rows

Triggers:

A trigger is a stored procedure in database which automatically invokes whenever a special event in the database occurs. For example, a trigger can be invoked when a row is inserted into a specified table or when certain table columns are being updated.

create trigger [trigger\_name]

[before | after]

{insert | update | delete}

on [table\_name]

[for each row]

[trigger\_body]

Transactions:

Transactions group a set of tasks into a single execution unit. Each transaction begins with a specific task and ends when all the tasks in the group successfully complete. If any of the tasks fail, the transaction fails. Therefore, a transaction has only two results: **success** or **failure**.

**1. BEGIN TRANSACTION:**It indicates the start point of an explicit or local transaction.

**Syntax:**

BEGIN TRANSACTION transaction\_name ;

**2. SET TRANSACTION:** Places a name on a transaction.

**Syntax:**

SET TRANSACTION [ READ WRITE | READ ONLY ];

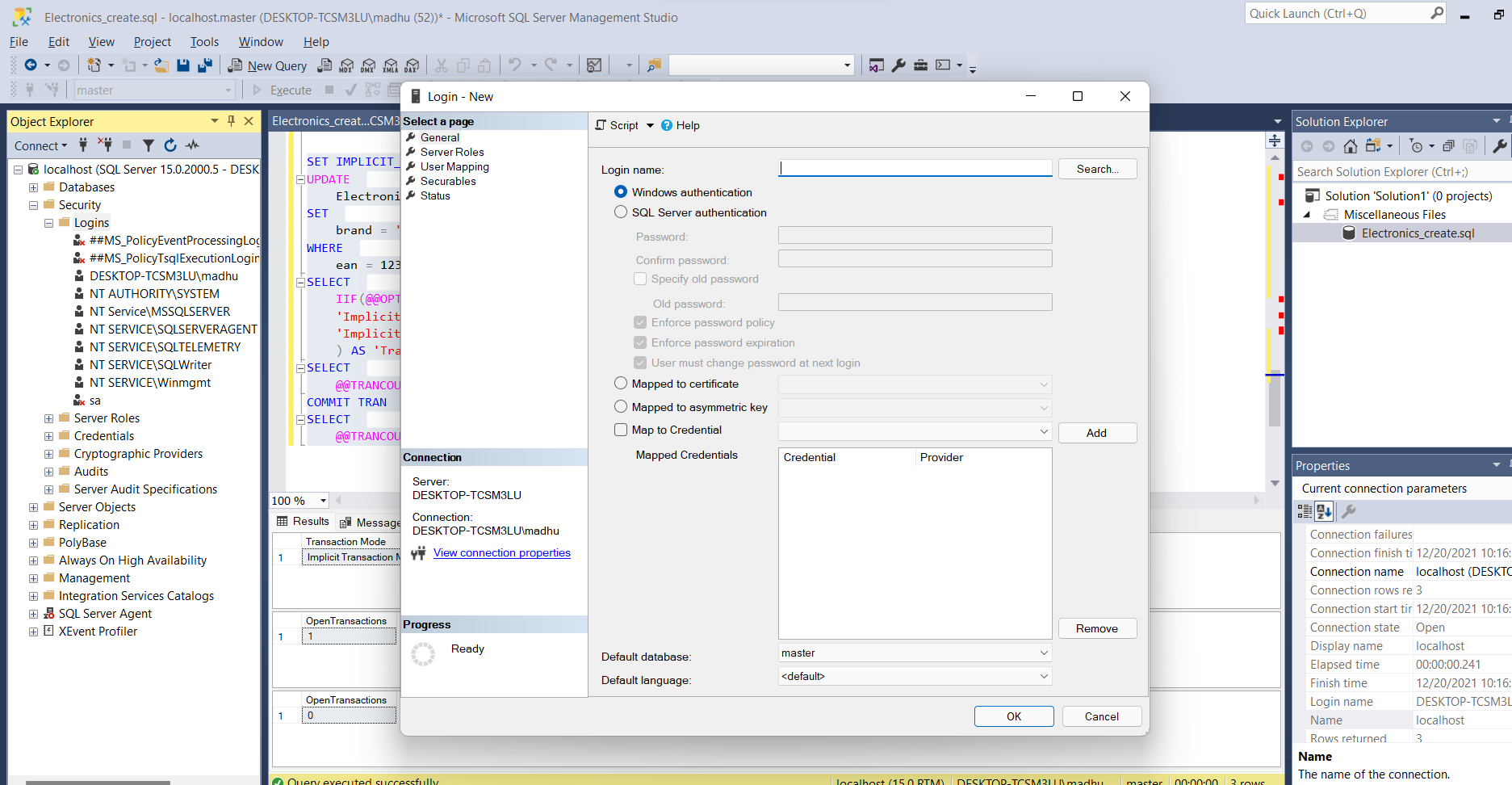
**3. COMMIT:** If everything is in order with all statements within a single transaction, all changes are recorded together in the database is called **committed**. The COMMIT command saves all the transactions to the database since the last COMMIT or ROLLBACK command. 

**Syntax:** 

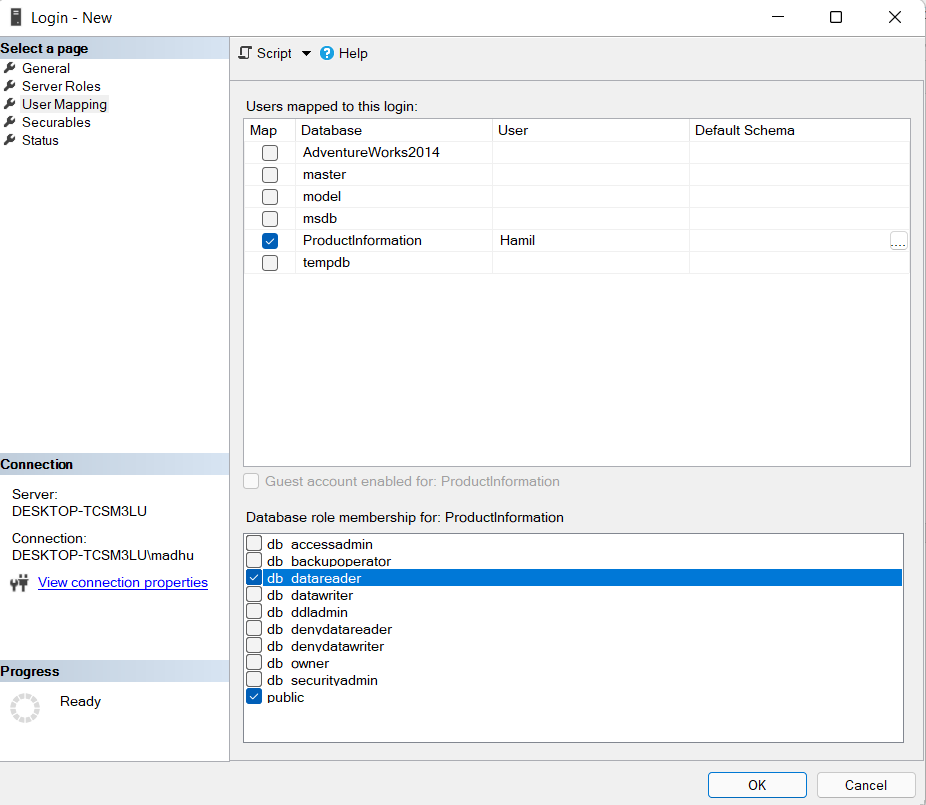
COMMIT;

## Security:

Security is the main concept in database in which securing the data that is being stored is the essential part in database concept and creating new users and granting required privileges to the existing users which can only be done by the administrator.

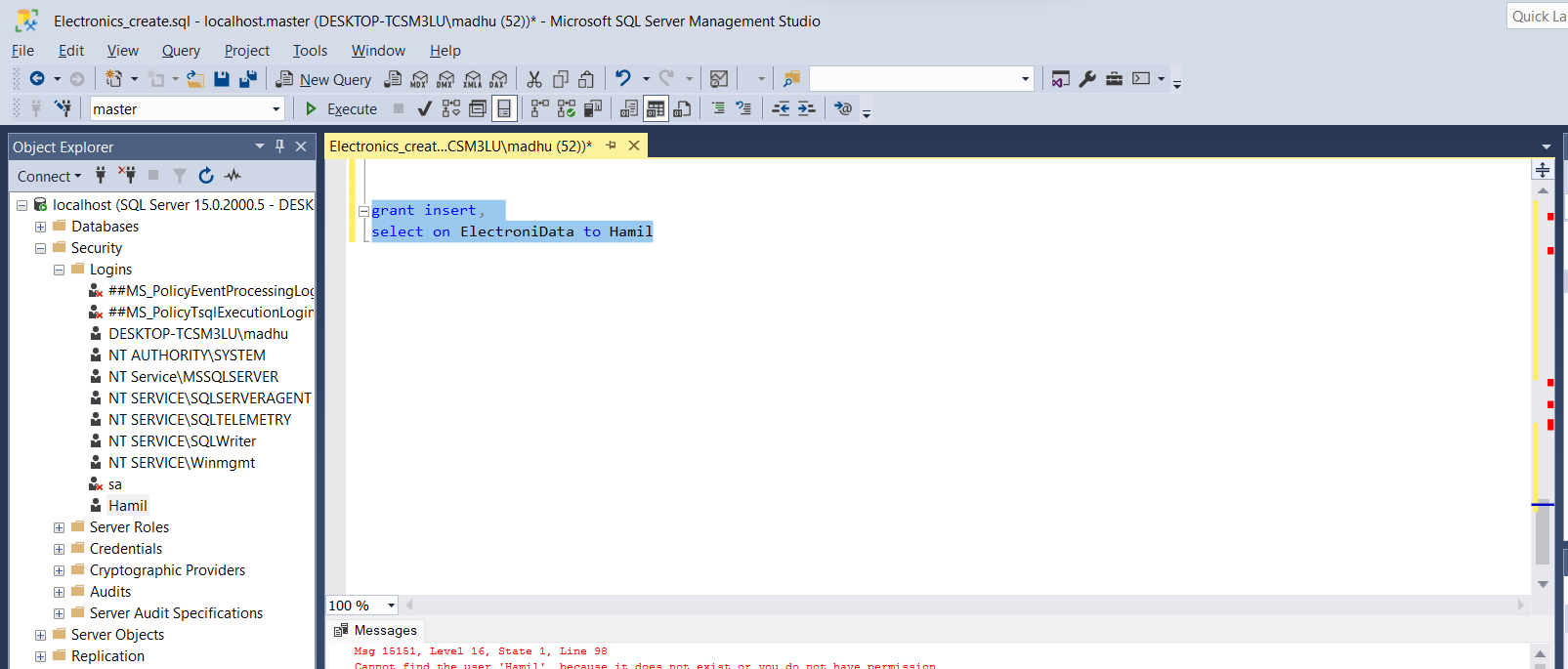


This screen refers to create a new user by selecting either the windows authentication or the sql server authentication. If we are selecting sql server authentication we need to provide the password for the said user that is going to be created. After mentioning the password move to user mapping role and select the database that is required to assign for newly created user. Along with the database mapping role mapping is also required with certain privileges.

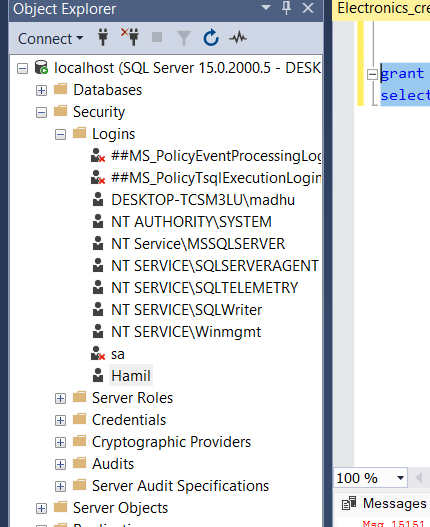


grant insert,

select on Countries to sa



Each database will be having required no of users who can maintain the database and their identity and mapping will be presented in the logins tab which is under security tab.



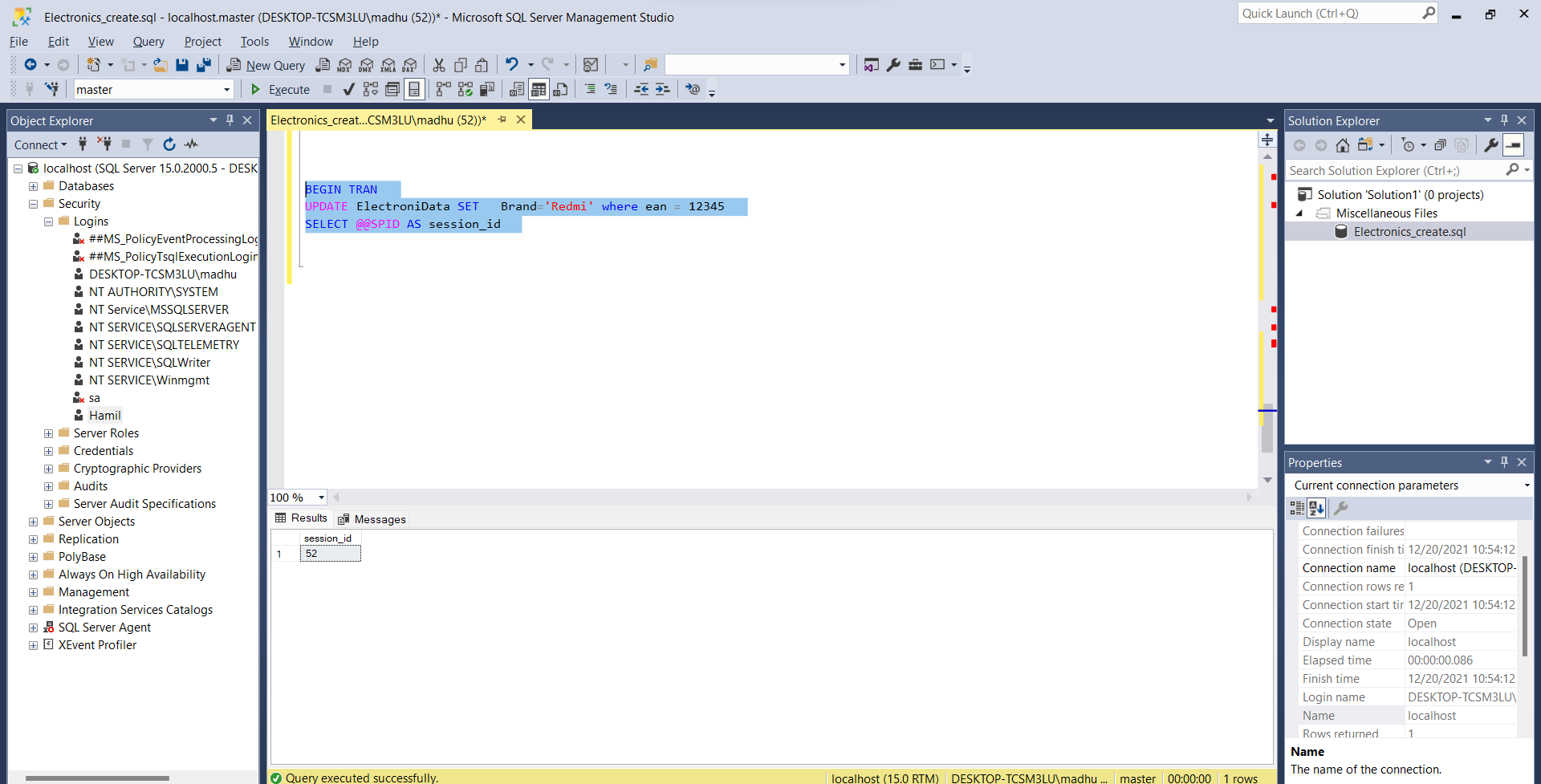
## Locking:

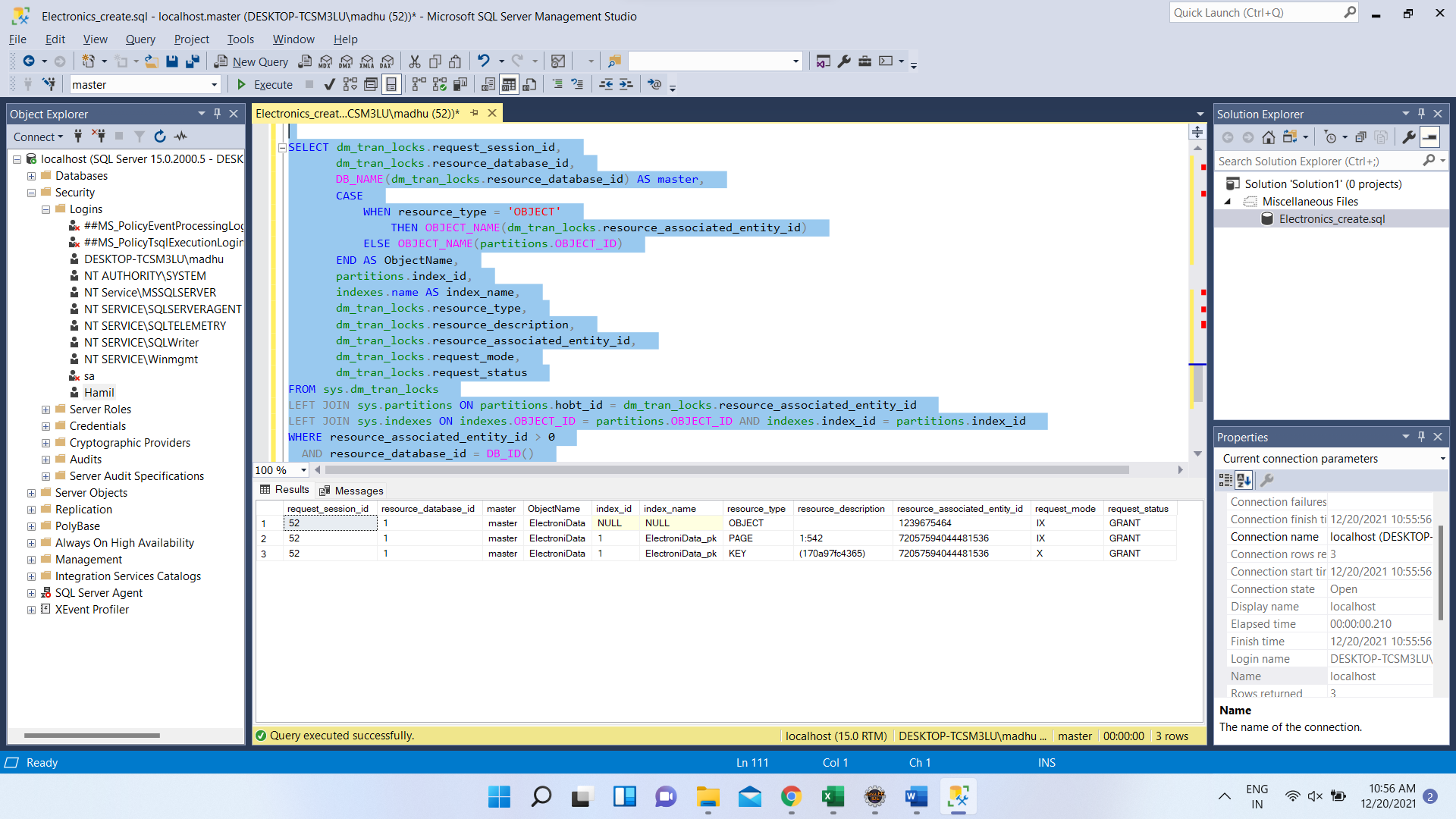
Applying Lock on the tables is a good practice in sql language because it locking on a table doesn’t allow user to perform any task and this will apply for all no of users who are going to use the said. This will reduce the concurrency and incorrect updation of the data which is being stored in the database. This is the mechanism which can also be applied to a row and a record that is particularly required.

BEGIN TRAN

UPDATE Countries SET state='Chicago' where stateid = 12345

SELECT @@SPID AS session\_id





SELECT dm\_tran\_locks.request\_session\_id,

dm\_tran\_locks.resource\_database\_id,

DB\_NAME(dm\_tran\_locks.resource\_database\_id) AS master,

CASE

WHEN resource\_type = 'OBJECT'

THEN OBJECT\_NAME(dm\_tran\_locks.resource\_associated\_entity\_id)

ELSE OBJECT\_NAME(partitions.OBJECT\_ID)

END AS ObjectName,

partitions.index\_id,

indexes.name AS index\_name,

dm\_tran\_locks.resource\_description,

dm\_tran\_locks.resource\_associated\_entity\_id,

dm\_tran\_locks.request\_mode,

dm\_tran\_locks.resource\_type,

dm\_tran\_locks.request\_status

FROM sys.dm\_tran\_locks

LEFT JOIN sys.partitions ON partitions.hobt\_id = dm\_tran\_locks.resource\_associated\_entity\_id

LEFT JOIN sys.indexes ON indexes.OBJECT\_ID = partitions.OBJECT\_ID AND indexes.index\_id = partitions.index\_id

WHERE resource\_associated\_entity\_id > 0

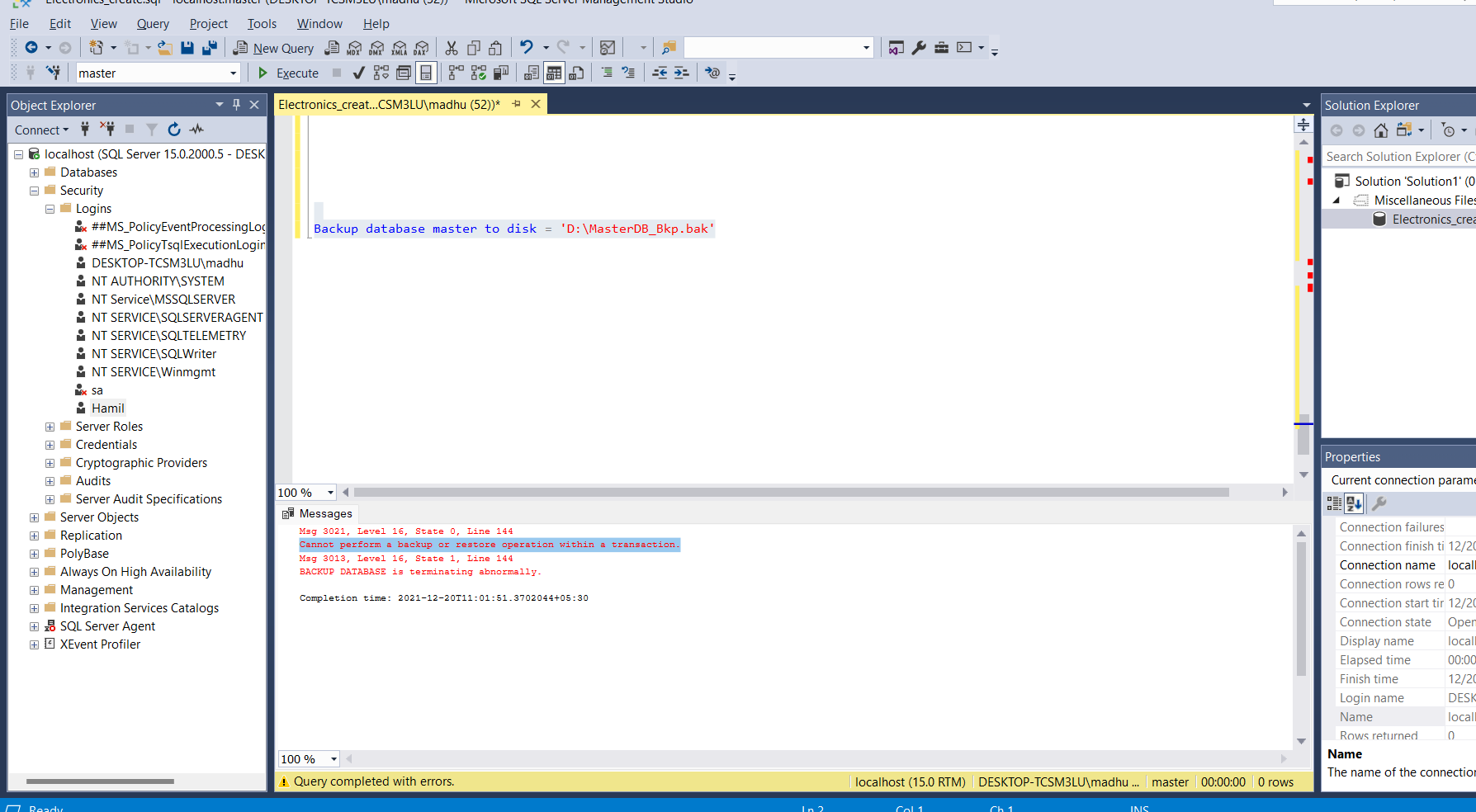
AND resource\_database\_id = DB\_ID()

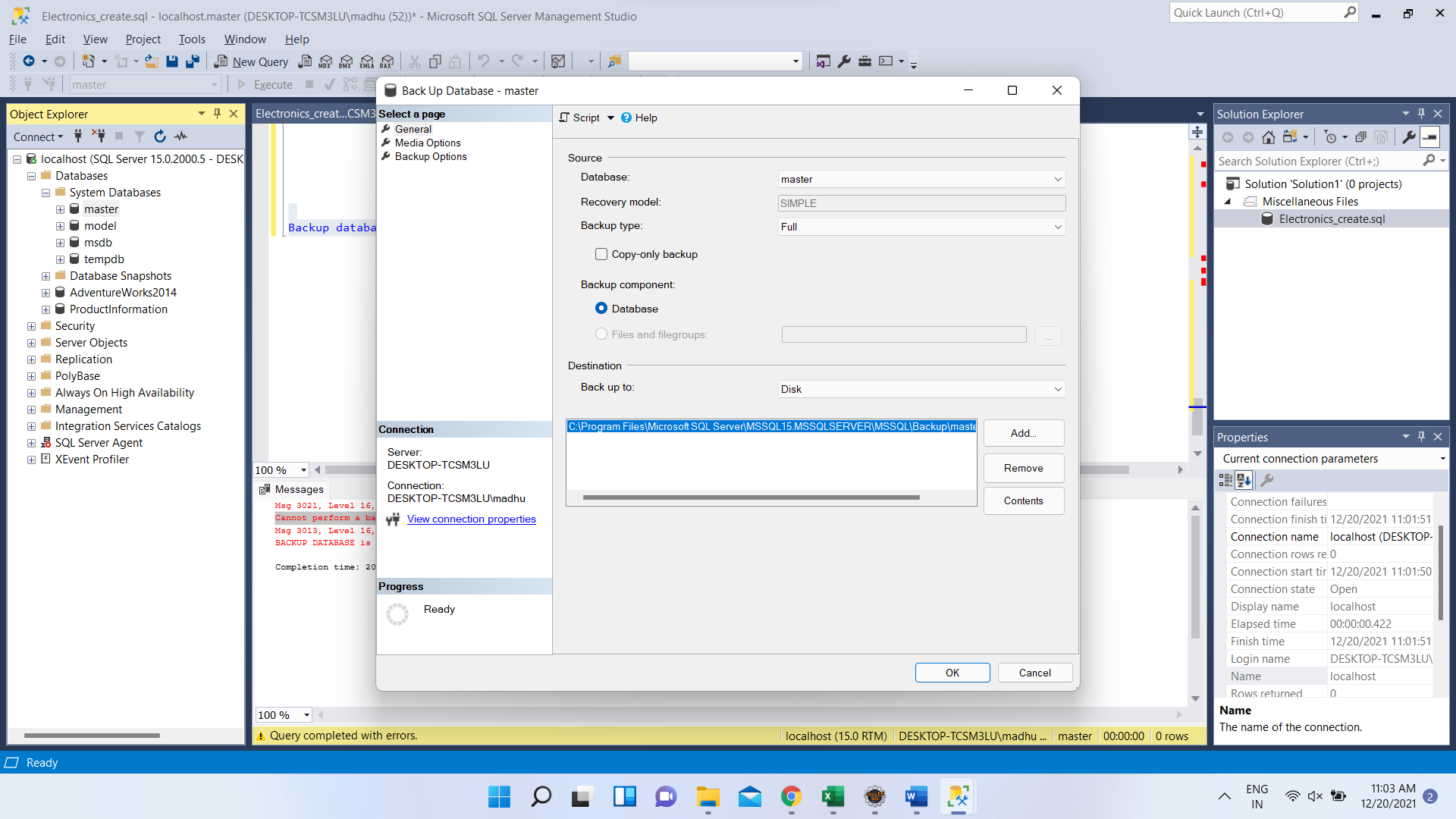
and request\_session\_id=52

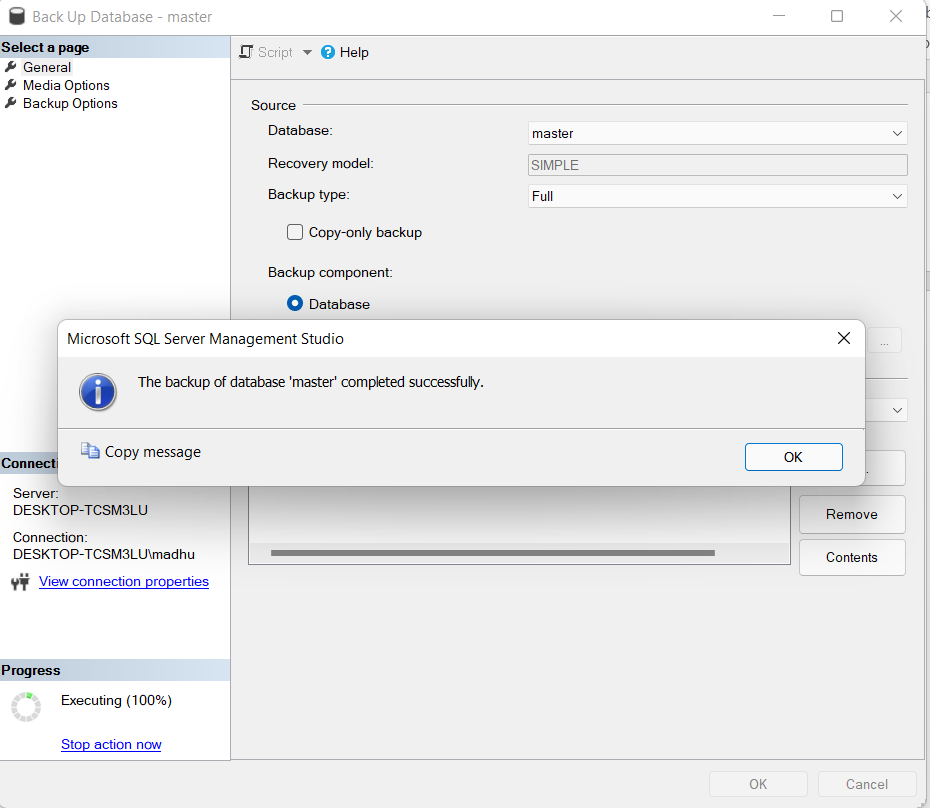
ORDER BY request\_session\_id, resource\_associated\_entity\_id

## Backup:

Performing backup conditions in database is an essential scope of work that is required to perform. We have multiple types of backups that are being performed in database are copy-only backup, data backup, database backup, differential backup, full backup, log backup, file backup, partial backup.







Python Programming:

import sqlite3

cnt = sqlite3.connect("backup.dp")

cnt.execute('''CREATE TABLE gfg(NAME TEXT,POINTS INTEGER,ACCURACY REAL);''')

cnt.execute('''INSERT INTO gfg VALUES(

'Count Inversion',20,80.5);''')

cnt.execute('''INSERT INTO gfg(ACCURACY, POINTS, NAME) VALUES(

90.5, 15, 'Kadanes Algo');''')

cnt.execute('''INSERT INTO gfg(NAME, ACCURACY, POINTS) VALUES(

'REVERSE STR', 100, 5);''')

cnt.commit()

print('Name, Points and Accuracy from '

      'records with accuracy greater than 85')

cursor = cnt.execute('''SELECT \* FROM gfg WHERE ACCURACY>85;''')

for i in cursor:

    print(i[0]+"    "+str(i[1])+"   "+str(i[2]))

print('')  # Print new line

print('Name, Accuracy from '

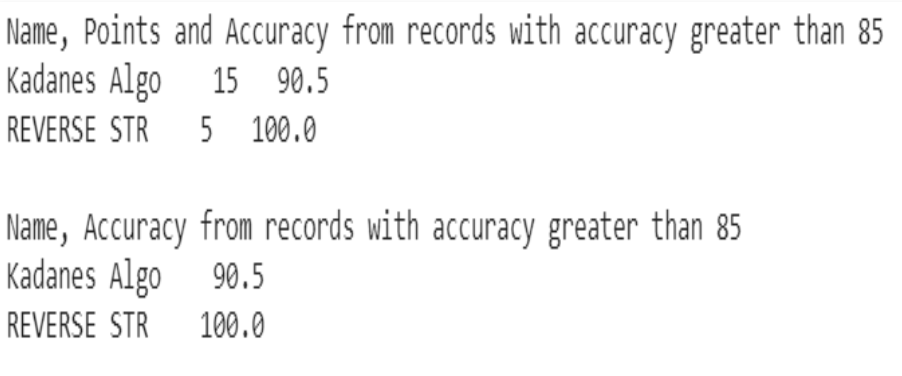
      'records with accuracy greater than 85')

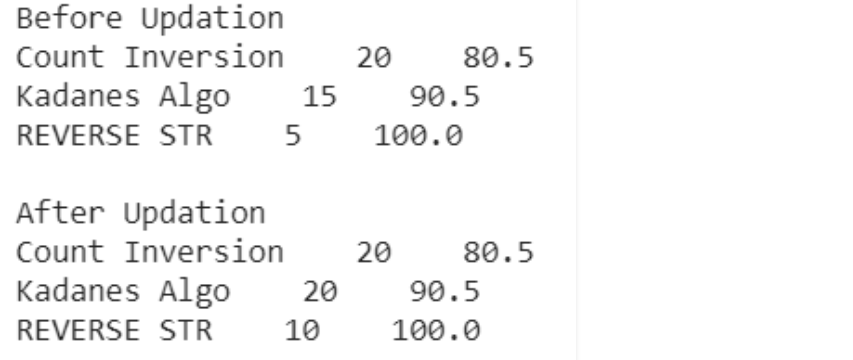
cursor = cnt.execute('''SELECT NAME, ACCURACY FROM

gfg WHERE ACCURACY>85;''')

for i in cursor:

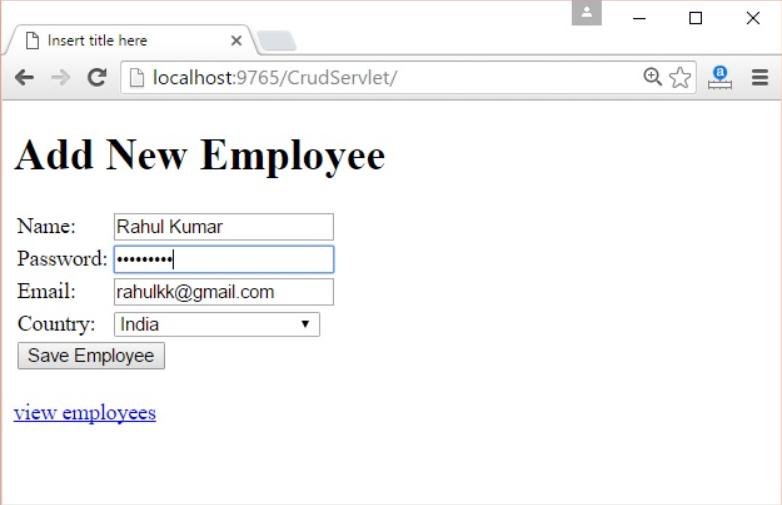
    print(i[0]+"    "+str(i[1]))

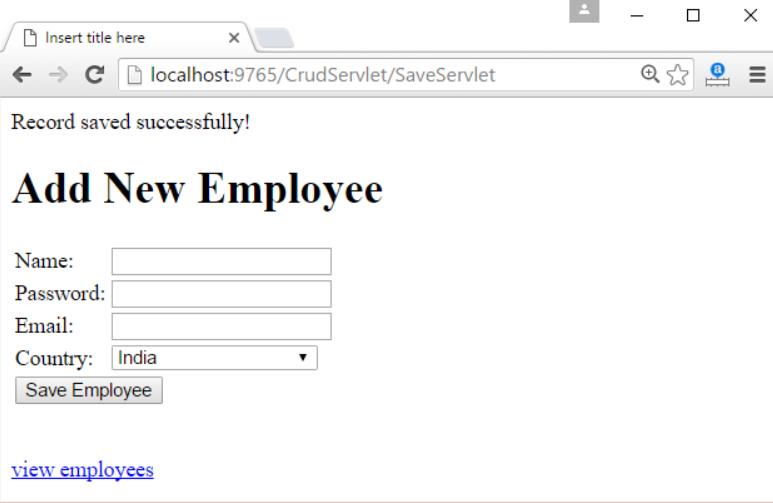
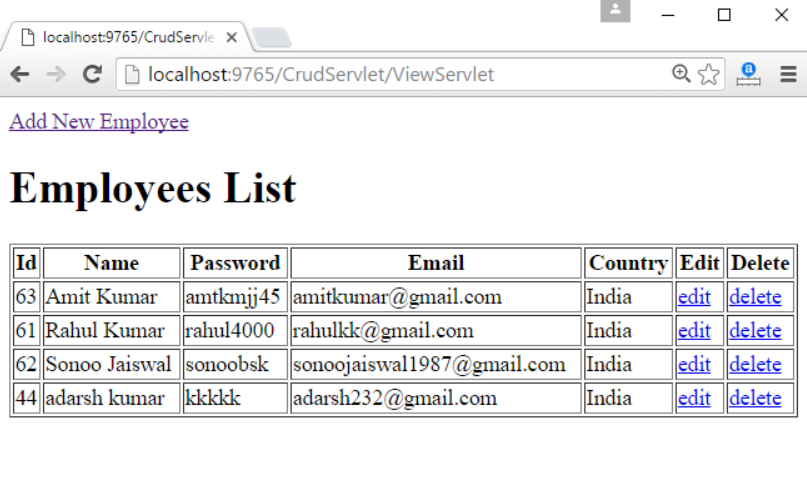


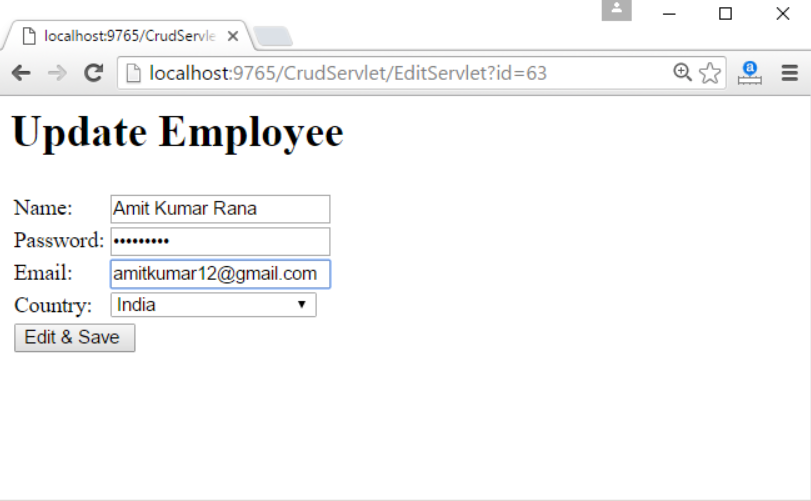


Java Program:

Working screenshots of the java code.





import java.util.\*;

import java.sql.\*;

public class EmpDao {

public static Connection getConnection(){

Connection con=null;

try{

Class.forName("oracle.jdbc.driver.OracleDriver");

con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","system","oracle");

}catch(Exception e){System.out.println(e);}

return con;

}

public static int save(Emp e){

int status=0;

try{

Connection con=EmpDao.getConnection();

PreparedStatement ps=con.prepareStatement(

"insert into user905(name,password,email,country) values (?,?,?,?)");

ps.setString(1,e.getName());

ps.setString(2,e.getPassword());

ps.setString(3,e.getEmail());

ps.setString(4,e.getCountry());

status=ps.executeUpdate();

con.close();

}catch(Exception ex){ex.printStackTrace();}

return status;

}

public static int update(Emp e){

int status=0;

try{

Connection con=EmpDao.getConnection();

PreparedStatement ps=con.prepareStatement(

"update user905 set name=?,password=?,email=?,country=? where id=?");

ps.setString(1,e.getName());

ps.setString(2,e.getPassword());

ps.setString(3,e.getEmail());

ps.setString(4,e.getCountry());

ps.setInt(5,e.getId());

status=ps.executeUpdate();

con.close();

}catch(Exception ex){ex.printStackTrace();}

return status;

}

public static int delete(int id){

int status=0;

try{

Connection con=EmpDao.getConnection();

PreparedStatement ps=con.prepareStatement("delete from user905 where id=?");

ps.setInt(1,id);

status=ps.executeUpdate();

con.close();

}catch(Exception e){e.printStackTrace();}

return status;

}

public static Emp getEmployeeById(int id){

Emp e=new Emp();

try{

Connection con=EmpDao.getConnection();

PreparedStatement ps=con.prepareStatement("select \* from user905 where id=?");

ps.setInt(1,id);

ResultSet rs=ps.executeQuery();

if(rs.next()){

e.setId(rs.getInt(1));

e.setName(rs.getString(2));

e.setPassword(rs.getString(3));

e.setEmail(rs.getString(4));

e.setCountry(rs.getString(5));

}

con.close();

}catch(Exception ex){ex.printStackTrace();}

return e;

}

public static List<Emp> getAllEmployees(){

List<Emp> list=new ArrayList<Emp>();

try{

Connection con=EmpDao.getConnection();

PreparedStatement ps=con.prepareStatement("select \* from user905");

ResultSet rs=ps.executeQuery();

while(rs.next()){

Emp e=new Emp();

e.setId(rs.getInt(1));

e.setName(rs.getString(2));

e.setPassword(rs.getString(3));

e.setEmail(rs.getString(4));

e.setCountry(rs.getString(5));

list.add(e);

}

con.close();

}catch(Exception e){e.printStackTrace();}

return list;

}

}

## Suggested Future Work:

After going through this database related concepts are pretty much interesting while getting to know and very much excited while working on such concepts and trying to implement the concepts on system with the help of one concept that is enterprise organization systems which has to organize their employee data perform some crud operations which are create, select, read, update and delete the data according the requirement needs. With the concept knowledge which we have we will be trying to working on normalizing the data and analysis of the data to perform the required report conclusions on any data and the data can be anything with which we can provide security using our database features and maintaining backup of each data that is being uploading into the database periodically and storing into the backup files on a hard storage functions which will provide more security of the data.

Activity Log:

|  |  |  |
| --- | --- | --- |
|  | Name | Topics Covered |
| Week 1 | Jagadeesh Kumar Periketi | Title Page |
| Rama Jaswanth Konakalla | Initial Proposal |
| Shiva Chembeti | Initial Proposal |

|  |  |  |
| --- | --- | --- |
|  | Name | Topics Covered |
| Week 2 | Jagadeesh Kumar Periketi | Data Sources |
| Rama Jaswanth Konakalla | Data Sources |
| Shiva Chembeti | Alternative Way to store data sources |

|  |  |  |
| --- | --- | --- |
|  | Name | Topics Covered |
| Week 3 | Jagadeesh Kumar Periketi | Alternative Way to store data sources |
| Rama Jaswanth Konakalla | Relational DB Design Process |
| Shiva Chembeti | Relational DB Design Process |

|  |  |  |
| --- | --- | --- |
|  | Name | Topics Covered |
| Week 4 | Jagadeesh Kumar Periketi | Relational DB Design Process |
| Rama Jaswanth Konakalla | Relational DB Design Process |
| Shiva Chembeti | DDL Scripts |

|  |  |  |
| --- | --- | --- |
|  | Name | Topics Covered |
| Week 5 | Jagadeesh Kumar Periketi | DDL Scripts |
| Rama Jaswanth Konakalla | DML Scripts |
| Shiva Chembeti | DML Scripts |

|  |  |  |
| --- | --- | --- |
|  | Name | Topics Covered |
| Week 6 | Jagadeesh Kumar Periketi | Indexes |
| Rama Jaswanth Konakalla | Views |
| Shiva Chembeti | Triggers |

|  |  |  |
| --- | --- | --- |
|  | Name | Topics Covered |
| Week 7 | Jagadeesh Kumar Periketi | Transactions |
| Rama Jaswanth Konakalla | Security |
| Shiva Chembeti | Locking, Backup |

|  |  |  |
| --- | --- | --- |
|  | Name | Topics Covered |
| Week 8 | Jagadeesh Kumar Periketi | Python, Java Programming |
| Rama Jaswanth Konakalla | Java Programming |
| Shiva Chembeti | Future Work |