Table of Contents

What is Database?

Database is a collection of information organized for easy access, management and maintenance.

- Examples:
  - Telephone directory
  - Customer data
  - Product inventory
  - Visitors' register
  - Weather records

Types of Data Models

- **Record based logical model**
  - Hierarchical data model
  - Network data model
  - Relational data model
- **Object based logical model**
  - Entity relationship model

DBMS Operations

- Adding new files
- Inserting data
- Retrieving data
- Modifying data
- Removing data
- Removing files

Advantages of DBMS

- Sharing of data across applications
- Enhanced security mechanism
- Enforce integrity constraints
- Better transaction support
- Backup and recovery features

Introduction to RDBMS

- A relational database refers to a database that stores data in a structured format, using rows and columns.
- This makes it easier to locate and access specific values within the database.
- It is "relational" because the values within each table are related to each other. Tables may also be related to other tables.
- The relational structure makes it possible to run queries across multiple tables at once.

Features of RDBMS

- Every piece of information is stored in the form of tables
- Has primary keys for unique identification of rows
- Has foreign keys to ensure data integrity
- Provides SQL for data access
- Uses indexes for faster data retrieval
- Gives access privileges to ensure data security

RDBMS VS TRADITIONAL APPROACH

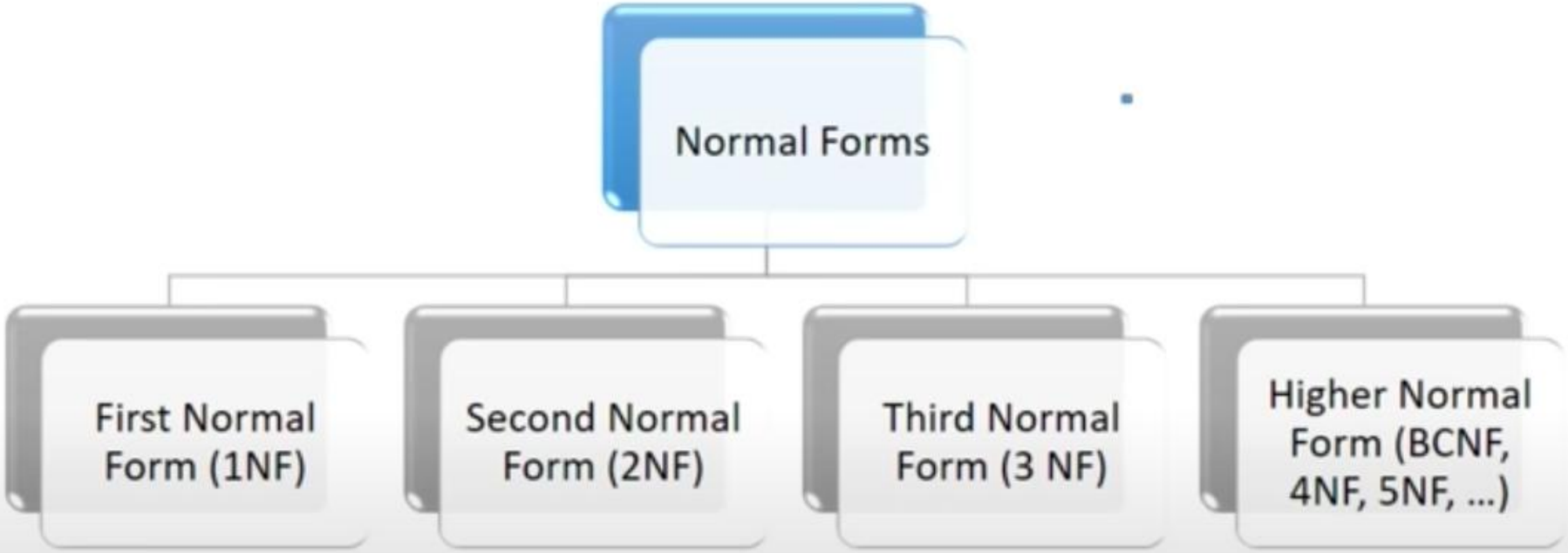
- The key difference is that RDBMS (relational database management system) applications store data in a tabular form, whereas in tradition approach, applications store data as files.

- There can be, but there will be no “relation” between the tables, like in a RDBMS. In traditional approach, data is generally stored in either a hierarchical form/navigational form. This means that a single data unit will have 0, 1 or more children nodes and 1 parent node.

Normalization

- Decompose larger, complex table into simpler and smaller ones
- Moves from lower normal forms to higher normal forms.

Normalization and Normal Forms



Need for Normalization

- In order to produce good database design
- To ensure all database operations to be efficiently performed
- Avoid any expensive DBMS operations
- Avoid unnecessary replication of information

Need for Normalization

- **RAW DATABASE**

Student_Details	Course_details		Pre-requisite	Result_details	
101 Jack 11/4/1975	M1 Advance Math	17	Basic Math	03/11/2015 82	A
102 Rock 10/04/1976	P4 Advance Physics	18	Basic Physics	22/11/2015 83	A
103 Mary 11/07/1975	B3 Advance Biology	10	Basic Biology	14/11/2015 68	B
104 Roby 10/04/1976	H6 Advance History	19	Basic History	22/11/2015 83	A
105 Jim 03/08/1978	C3 Advance Chemistry	12	Basic Biology	15/11/2015 50	C

Functional Dependency

- Consider the relation
  - Result (Student#, Course#, CourseName#, Marks#, Grade#)
    - Student# and course# together defines exactly one value of marks. Student#, course# ,Marks
    - Student# and course# determines Marks or Marks is functionally dependent on student# and course#
- Other functional dependencies in the relation:
  - Course# - CourseName
  - Marks# - Grade

Functional Dependency

- In a given relation R, P and Q are attributes. Attribute Q is functionally dependent on attribute P if each value of P determines exactly one value of Q.

Functional Dependency Types

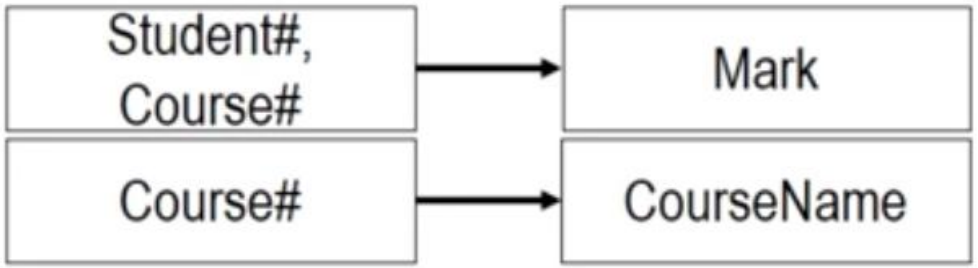
Partial Functional Dependency

Transitive Dependency

Functional Dependency Types

Partial Functional Dependency

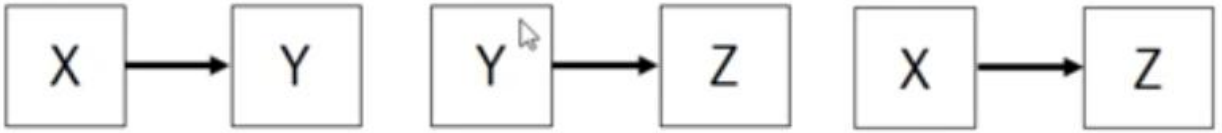
- Attribute Q is partially dependent on attribute P, if and only if it is dependent on the subset of attribute P.
- REPORT (Student#, Course#, StudentName, CourseName, Marks, Grade)



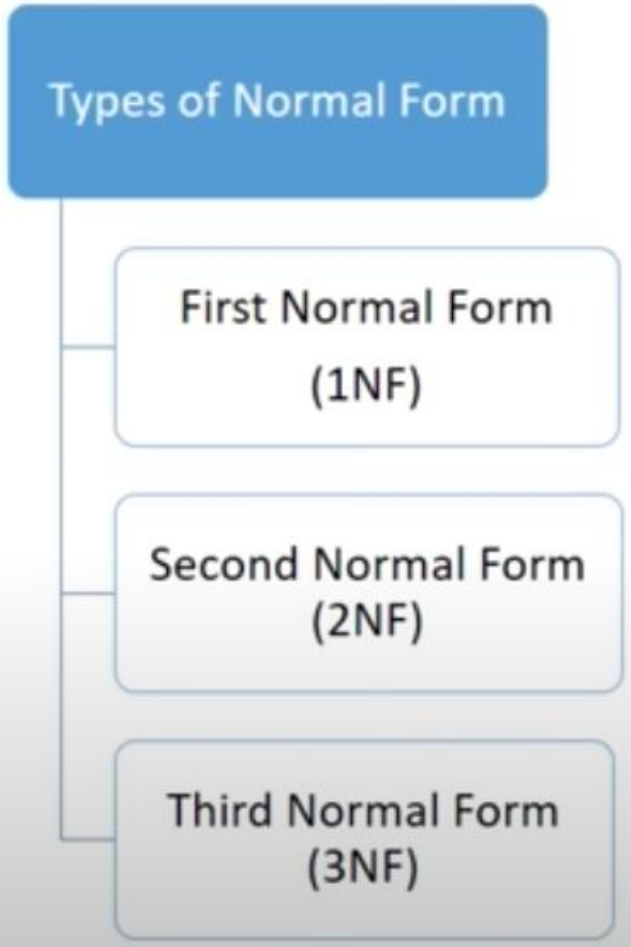
Functional Dependency Types

Transitive Dependency

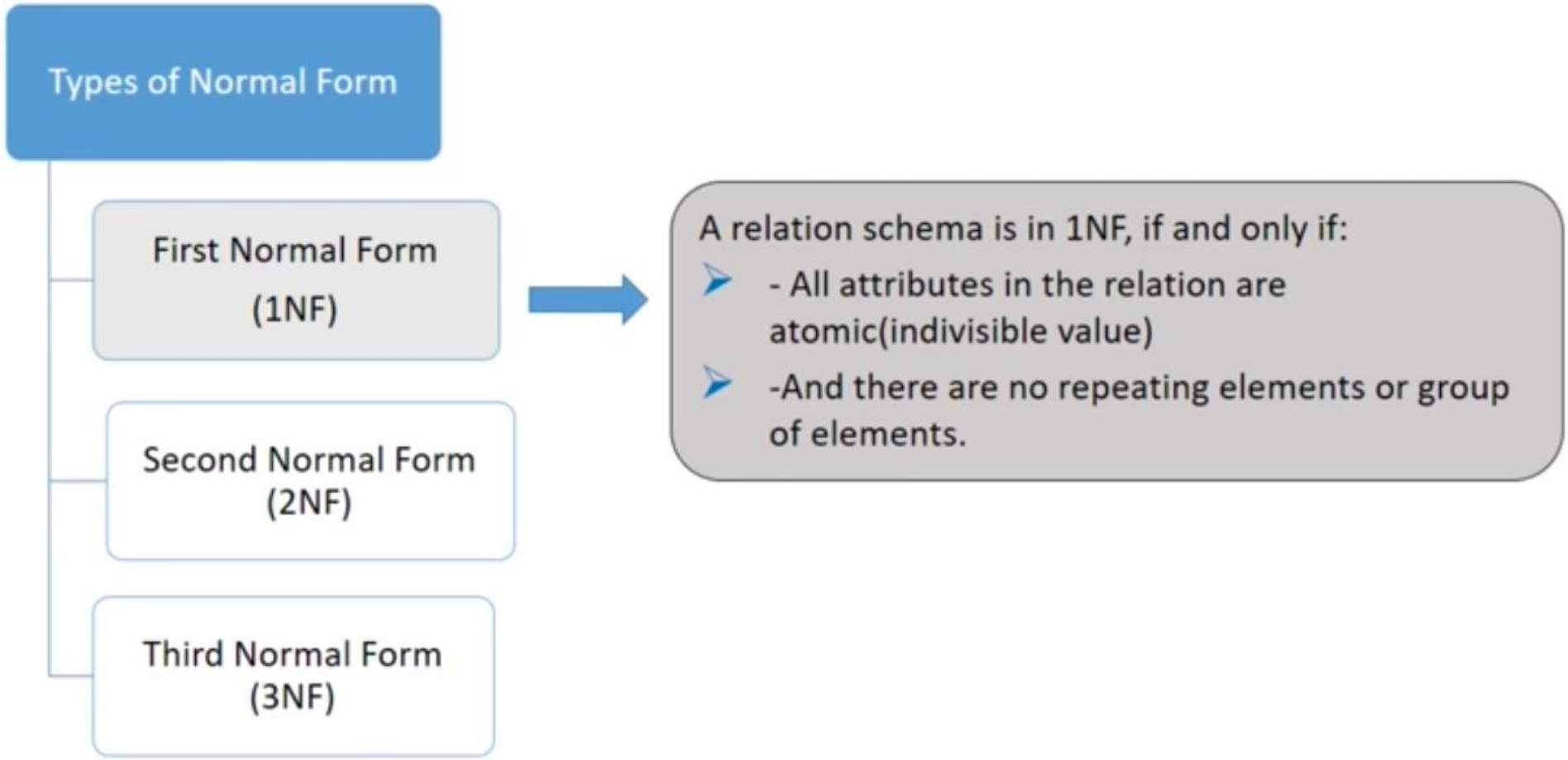
X, Y, Z are three attributes



Normalization



First Normal Form – (1NF)



First Normal Form – (1NF)

Student Marks Table

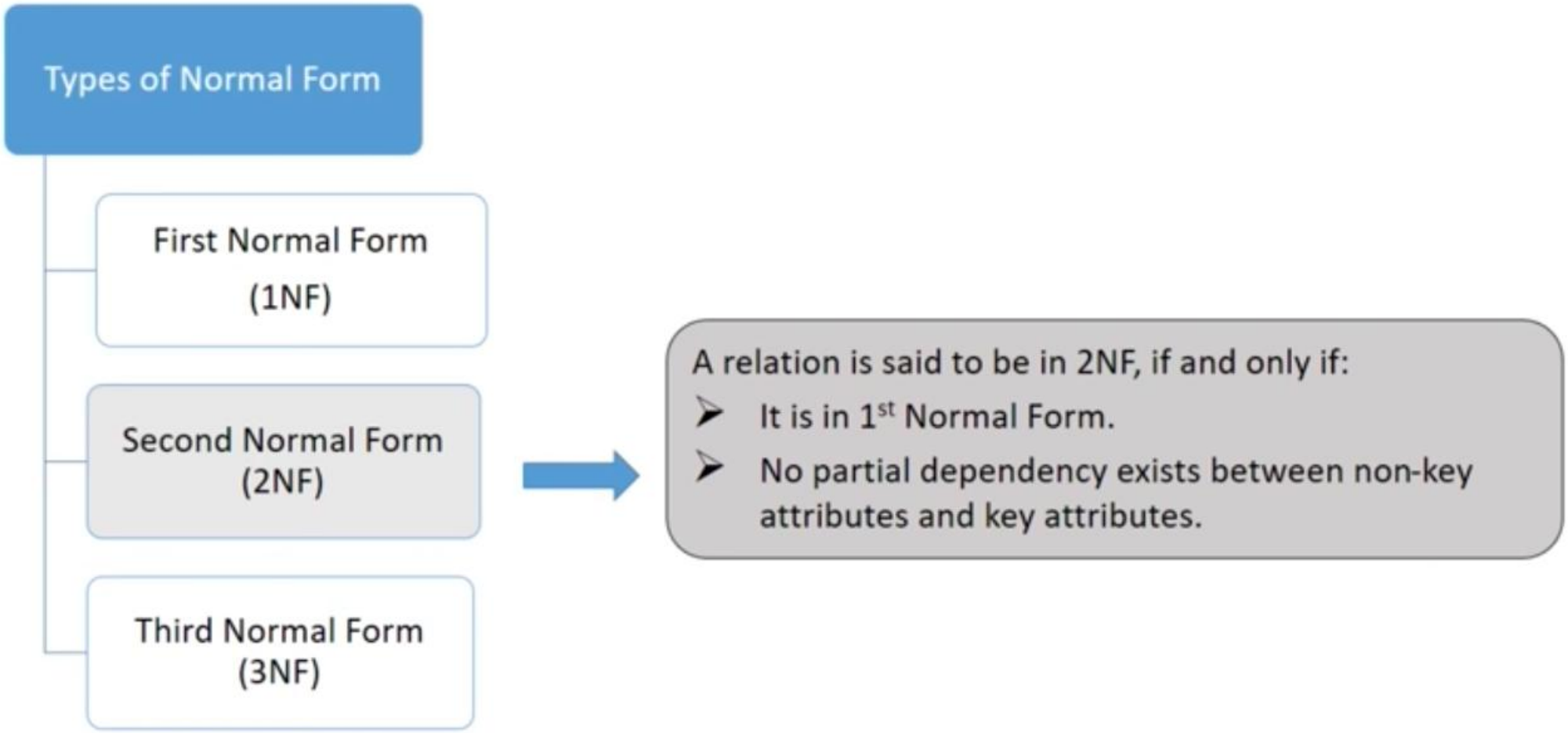
Student_Details	Course_details		Pre-requisite	Result_details	
101 Jack 11/4/1975	M1 Advance Math	17	Basic Math	03/11/2015 82	A
102 Rock 10/04/1976	P4 Advance Physics	18	Basic Physics	21/11/2015 83	A
103 Mary 11/07/1975	B3 Advance Biology	10	Basic Biology	12/11/2015 68	B
104 Roby 10/04/1976	H6 Advance History	19	Basic History	21/11/2015 83	A
105 Jim 03/08/1978	C3 Advance Chemistry	12	Basic Biology	12/11/2015 50	C

First Normal Form - (1NF)

Student Marks Table in 1NF

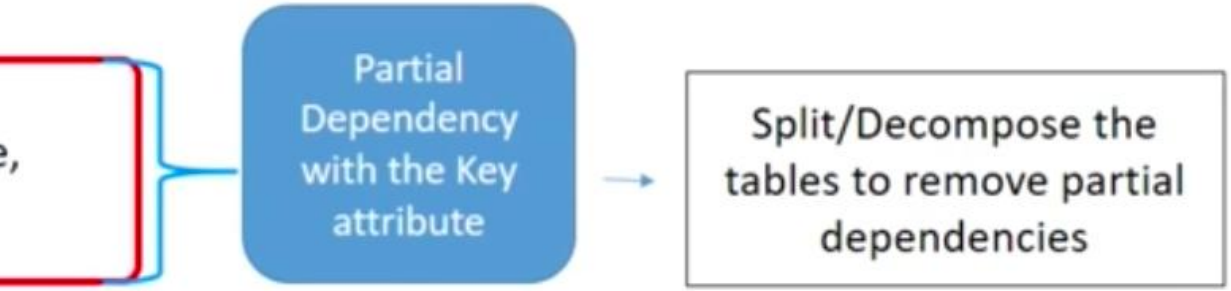
Student #	Student_ Name	Date Of Birth	Course#	CourseName	Pre-Requisite	Duration in days	Date Of Exam	Marks	Grade
101	Jack	11/4/1975	M1	Advance Math	Basic Math	17	02/11/ 2015	82	A
102	Roby	10/04/1976	P4	Advance Physics	Basic Physics	18	21/11/ 2015	83	A
103	Mary	11/07/1975	B3	Advance Biology	Basic Biology	10	12/11/ 2015	68	B
...	...	...	...	...	...	...	...	...	...

Second Normal Form – (2NF)



Second Normal Form – (2NF)

- Student# ,Course# → Marks
- Student#, Course# → Grade
- Marks → Grade
- Student# → StudentName, DOB
- Course# → CourseName, Pre-Requisite,
- DurationDays, Date of exam



Second Normal Form – (2NF)

Student Table

Student#	Student_Name	Date Of Birth
101	Jack	11/4/1975
102	Roby	10/04/1976
103	Mary	11/07/1975

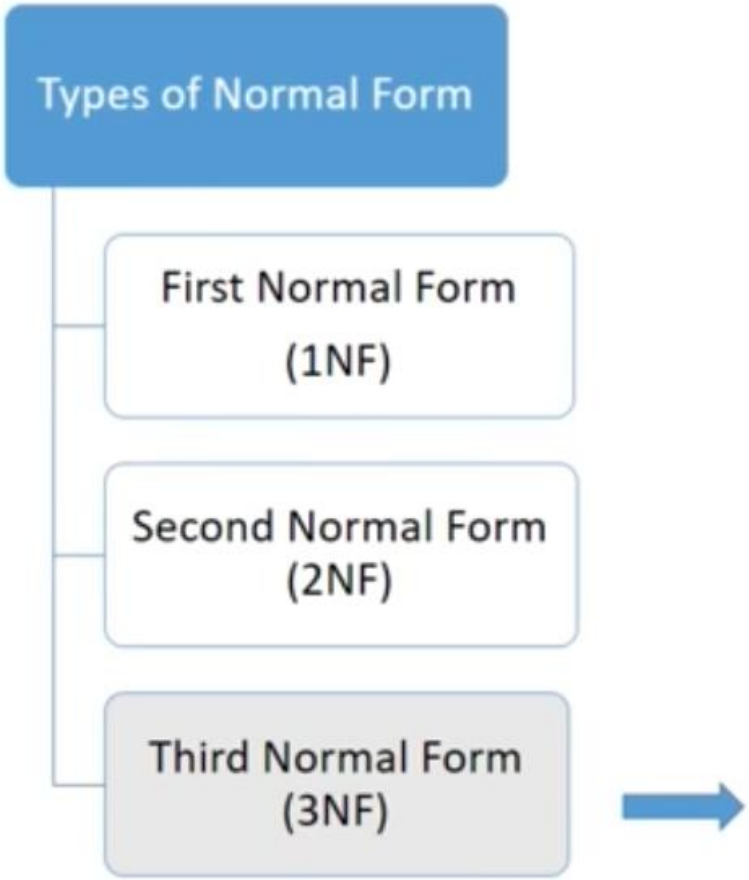
Result Table

Student#	Course#	Marks	Grade
101	M1	82	A
102	P4	83	A
103	B3	68	B

Course Table

Course#	CourseName	Prerequisite	Durationindays	Date Of Exam
M1	Advance Maths	Basic Math	17	02/11/2015
P4	Advance Physics	Basic Physics	18	21/11/2015
B3	Advance Biology	Basic Biology	10	12/11/2015

Third Normal Form – (3NF)



- A relation R is said to be in 3NF if and only if:
  - It is in 2NF.
  - No transitive dependency exists between non-key attributes and key attributes through another non-key attribute.

Third Normalization – (3NF)

Result Table

Student#	Course#	Marks
101	M1	82
102	P4	83
103	B3	68

Marks Grade Table

Marks	Grade
82	A
83	A
68	B

What is SQL ?

Programming language specifically designed for working with Database to...

- CREATE
- MANIPULATE
- SHARE/ACCESS

Why SQL?

SQL is widely popular because it offers the following advantages:

- Allows users to communicate i.e, access and manipulate the database.
- Allows users to retrieve data from a database.
- Allows users to create, update, modify and delete the database

SQL is a language for defining the structure of a database.

SQL Terms

Data

Data is defined as facts or figures, or information that's stored in or used by a computer.

Database

A database is a organized collection of data/information so that it can be easily accessed, managed and updated.

SQL Data Types

1. Numeric – bit, tinyint, smallint, int, bigint, decimal, numeric, float, real
2. Character/String - Char, Varchar, Text
3. Date/Time - Date, Time, Datetime, Timestamp, Year
4. Miscellaneous- Json, XML

SQL Constraints

Constraint	Description
Not Null	Ensures that a column does not have a NULL value.
Default	Provides a default value for a column when none is specified.
Unique	Ensures that all the values in a column are different.
Primary	Identifies each row/record in a database table uniquely.
Check	Ensures that all values in a column satisfy certain conditions.
Index	Creates and retrieves data from the database very quickly.

Subsets of SQL

SQL Command Groups

- **DDL** (Data Definition Language) : creation of objects
- **DML** (Data Manipulation Language) : manipulation of data
- **DCL** (Data Control Language) : assignment and removal of permissions
- **TCL** (Transaction Control Language) : saving and restoring changes to a database

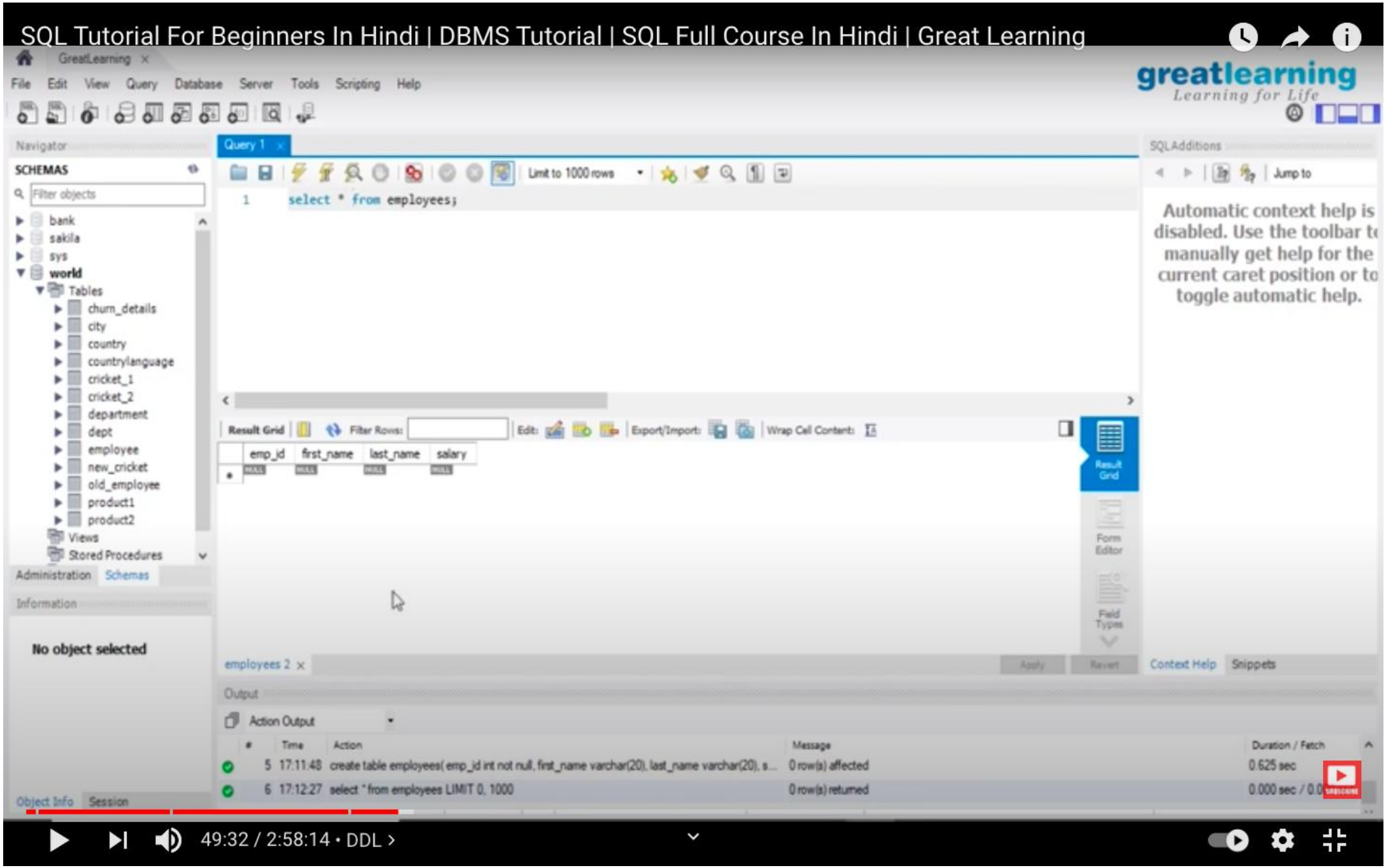
DDL – Data Definition Language

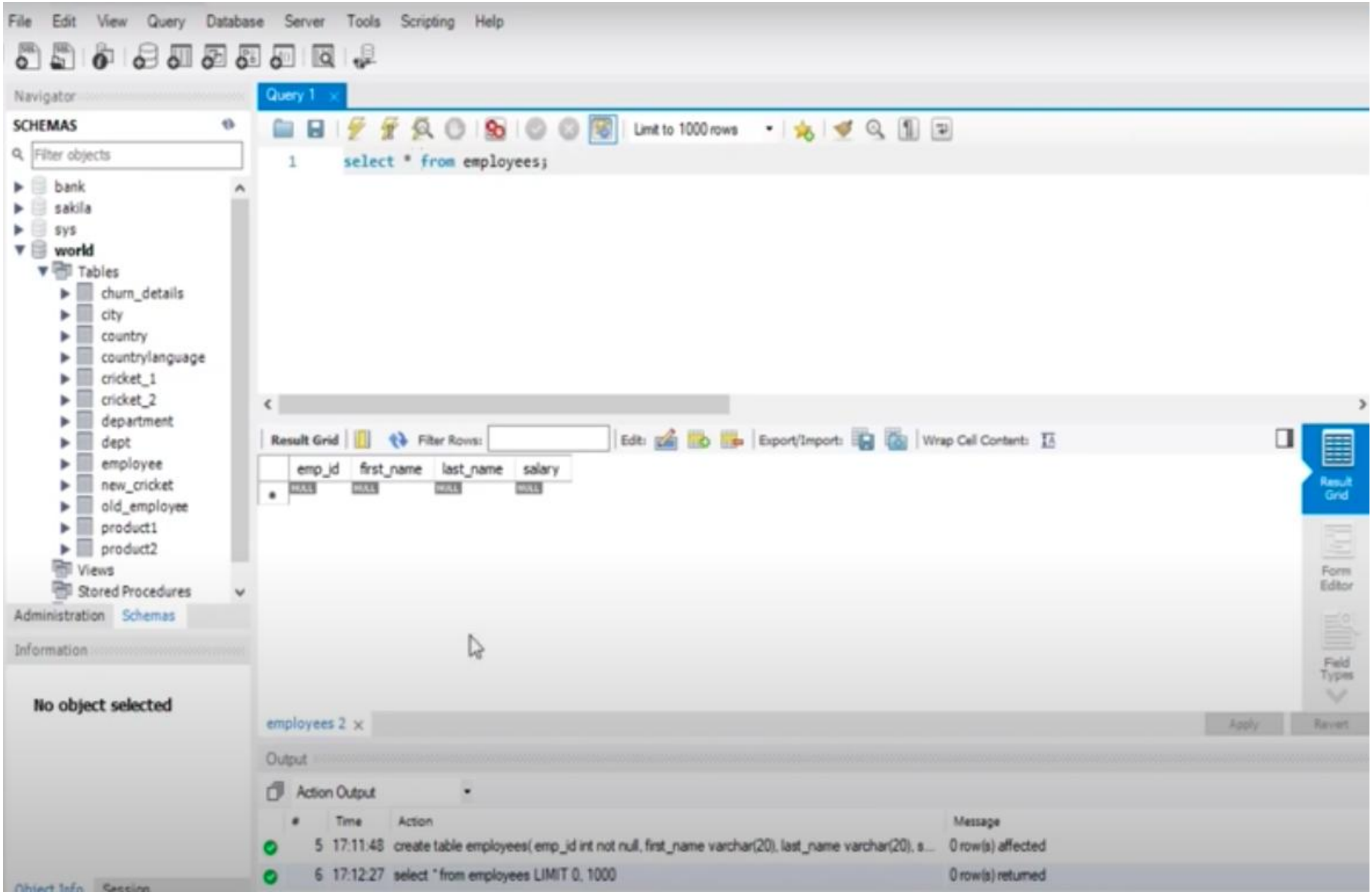
Command	Description
Create	Creates objects in the database/database objects
Alter	Alters the structures of the database/ database objects
Drop	Deletes objects from the database
Truncate	Removes all records from a table permanently
Rename	Renames an object

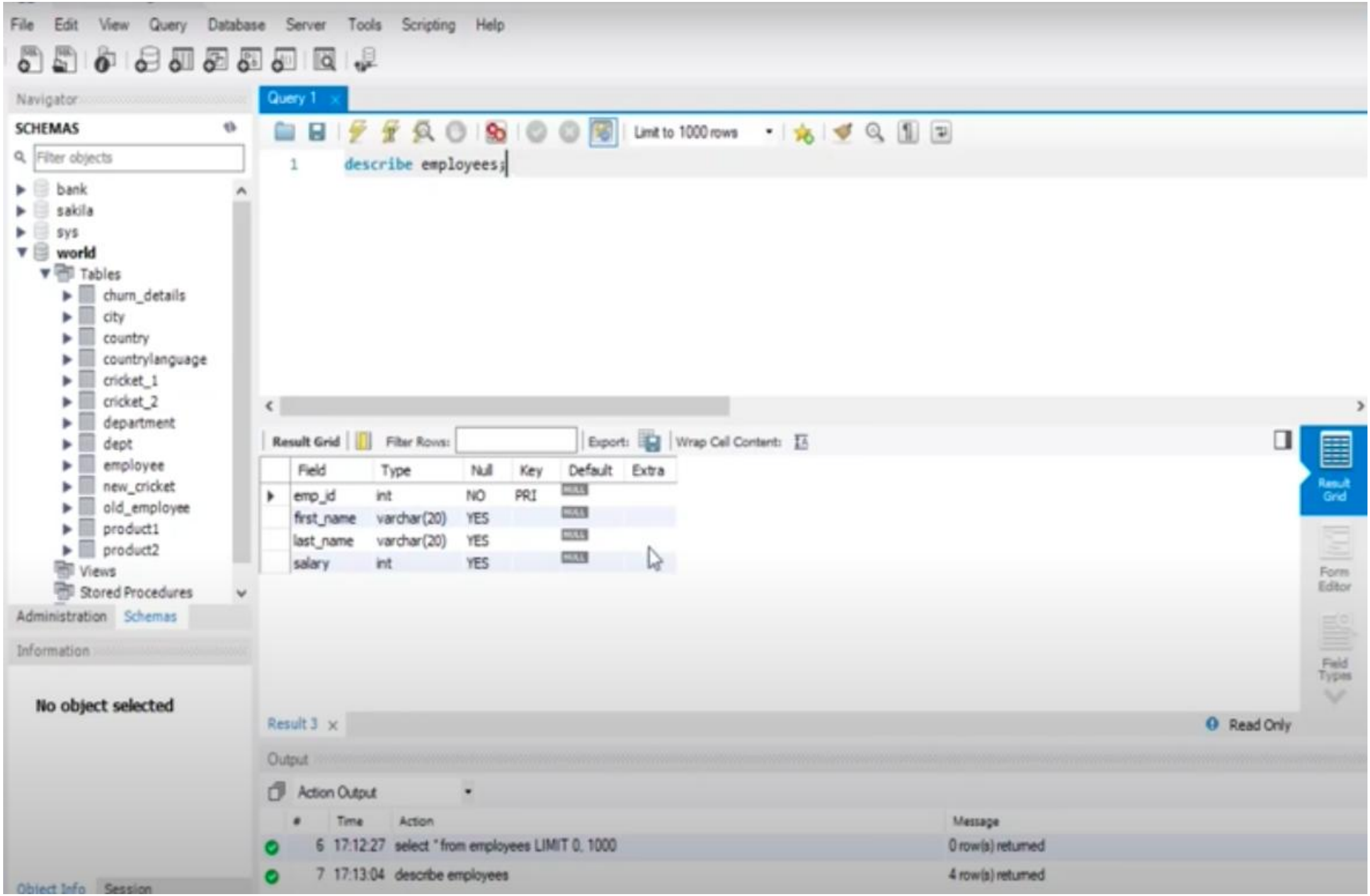
DDL - Data Definition Language – Create Command

```
CREATE TABLE employees (
  emp_id INT (10) NOT NULL,
  first_name VARCHAR(20),
  last_name VARCHAR(20) NOT NULL,
  salary int(10) NOT NULL,
  PRIMARY KEY (emp_id));
```

emp_id	first_name	last_name	salary


SQL Tutorial For Beginners In Hindi | DBMS Tutorial | SQL Full Course In Hindi | Great Learning
GreatLearning
File Edit View Query Database Server Tools Scripting Help
greatlearning
Learning for Life
Navigator
SCHEMAS
Filter objects
bank
sakila
sys
world
Tables
churn_details
city
country
countrylanguage
cricket_1
cricket_2
department
dept
employee
new_cricket
old_employee
product1
product2
Views
Stored Procedures
Administration Schemas
Information
No object selected
Object Info Session
Query 1
Limit to 1000 rows
1 select * from employees;
Result Grid Filter Rows: Edit: Export/Import: Wrap Cell Content:
emp_id first_name last_name salary
Result Grid
Form Editor
Field Types
employees 2
Apply Revert
SQLAdditions
Jump to
Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.
Context Help Snippets
Output
Action Output
Time Action Message Duration / Fetch
5 17:11:48 create table employees(emp_id int not null, first_name varchar(20), last_name varchar(20), s... 0 row(s) affected 0.625 sec
6 17:12:27 select * from employees LIMIT 0, 1000 0 row(s) returned 0.000 sec / 0.0
49:32 / 2:58:14 • DDL ›


File Edit View Query Database Server Tools Scripting Help
Navigator
SCHEMAS
Filter objects
bank
sakila
sys
world
Tables
churn_details
city
country
countrylanguage
cricket_1
cricket_2
department
dept
employee
new_cricket
old_employee
product1
product2
Views
Stored Procedures
Administration Schemas
Information
No object selected
Object Info Session
Query 1
Limit to 1000 rows
1 select * from employees;
Result Grid Filter Rows: Edit: Export/Import: Wrap Cell Content:
emp_id first_name last_name salary
NULL NULL NULL NULL
Result Grid
Form Editor
Field Types
employees 2
Apply Revert
Output
Action Output
Time Action Message
5 17:11:48 create table employees(emp_id int not null, first_name varchar(20), last_name varchar(20), s... 0 row(s) affected
6 17:12:27 select * from employees LIMIT 0, 1000 0 row(s) returned


File Edit View Query Database Server Tools Scripting Help
Navigator
SCHEMAS
Filter objects
bank
sakila
sys
world
Tables
churn_details
city
country
countrylanguage
cricket_1
cricket_2
department
dept
employee
new_cricket
old_employee
product1
product2
Views
Stored Procedures
Administration Schemas
Information
No object selected
Object Info Session
Query 1
Limit to 1000 rows
1 describe employees;
Result Grid Filter Rows: Export: Wrap Cell Content:
Field Type Null Key Default Extra
emp_id int NO PRI NULL
first_name varchar(20) YES NULL
last_name varchar(20) YES NULL
salary int YES NULL
Result Grid
Form Editor
Field Types
Result 3
Read Only
Output
Action Output
Time Action Message
6 17:12:27 select * from employees LIMIT 0, 1000 0 row(s) returned
7 17:13:04 describe employees 4 row(s) returned

DDL - Data Definition Language – Alter Command

```
ALTER TABLE employees ADD COLUMN contact
INT(10);
```

emp_id	first_name	last_name	salary	contact
101	Steven	Cohen	10000	
102	Edwin	Thom	15000	
103	Harry	Potter	20000	

DDL - Data Definition Language – Rename Command

```
ALTER TABLE employees RENAME
COLUMN contact TO job_code;
```

empl_id	first_name	last_name	salary	job_code
101	Steven	Cohen	10000	
102	Edwin	Thomas	15000	
103	Harry	Potter	20000	

DDL - Data Definition Language – Truncate Command

```
TRUNCATE TABLE employees;
```

emp_id	first_name	last_name	salary
101	Steven	Cohen	10000
102	Edwin	Thomas	15000
103	Harry	Potter	20000

DDL - Data Definition Language – Drop Command

```
DROP TABLE table_name;

DROP TABLE employees;
```

emp_id	first_name	last_name	salary
101	Steven	Cohen	10000
102	Edwin	Thomas	15000
103	Harry	Potter	20000

DML – Data Manipulation Language

Command	Description
Insert	Insert data into a table
Update	Updates existing data within a table
Delete	Deletes specified/all records from a table

DML – Data Manipulation Language – INSERT Command

```
INSERT INTO employees
(emp_id,first_name,last_name,salary) VALUES
(101, 'Steven', 'King', 10000);
```

```
INSERT INTO employees
(emp_id,first_name,last_name,salary) VALUES
(102, 'Edwin', 'Thomas', 15000 );
```

```
INSERT INTO employees
(emp_id,first_name,last_name,salary) VALUES
(103, 'Harry', 'Potter', 20000);
```

emp_id	first_name	last_name	salary
101	Steven	King	10000
102	Edwin	Thomas	15000
103	Harry	Potter	20000

DML – Data Manipulation Language – UPDATE Command

```
UPDATE employees
SET last_name='Cohen'
WHERE emp_id=101;
```

emp_id	first_name	last_name	salary
101	Steven	Cohen	10000
102	Edwin	Thomas	15000
103	Harry	Potter	20000

DML – Data Manipulation Language - DELETE Command

```
DELETE FROM employees WHERE emp_id IN
(101,103);
```

emp_id	first_name	last_name	salary
101	Steven	Cohen	10000
102	Edwin	Thomas	15000
103	Harry	Potter	20000

DCL – Data Control Language

Command	Description
Grant	Gives access privileges to database
Revoke	Withdraws access privileges given with the grant command

```
GRANT <Privilege list> ON
<Relation Name> TO
<USER>
```

```
REVOKE <Privilege list> ON
<Relation Name> TO
<USER>
```

TCL – Transaction Control

Command	Description
Commit	Saves the work done
Rollback	Restores database to origin state since the last commit
Savepoint	Identify a point in a transaction to which you can roll back later

SQL Operators

SQL Operators - Filter

WHERE Clause :

- Used to specify a condition while fetching the data from a single table or by joining with multiple tables.
- Not only used in the SELECT statement, but it is also used in the UPDATE, DELETE statement, etc.,

emp_id	first_name	last_name	salary
101	Steven	Cohen	10000
102	Edwin	Thomas	15000
103	Harry	Potter	20000

e.g.

```
SELECT * FROM employees WHERE
emp_id=101;
```

emp_id	first_name	last_name	salary
101	Steven	Cohen	10000

The example mentioned above extracts all the columns from the table 'employees' whose emp_id=101

SQL Operators – Logical

Operator	IllustrativeExample	Result
AND	(5<2) AND (5>3)	FALSE
OR	(5<2) OR (5>3)	TRUE
NOT	NOT(5<2)	TRUE

Sample Queries:

```
SELECT * FROM employees WHERE first_name = 'Steven' and salary = 15000;

SELECT * FROM employees WHERE first_name = 'Steven' OR salary =15000;

SELECT * FROM employees WHERE first_name = 'Steven' and salary !=10000;
```

SQL Operators – Comparison

Comparison Operators	
Symbol	Meaning
=	Equal to
>	Greater than
>=	Greater than or equal to
<	Less than
<=	Less than or equal to
<> or !=	Not equal to

Sample Queries:

```
SELECT * FROM employees WHERE first_name = 'Steven'
AND salary <=10000;

SELECT * FROM employees WHERE first_name = 'Steven'
OR salary >=10000;
```

SQL Operators – Special

Special Operators	
BETWEEN	Checks an attribute value within range
LIKE	Checks an attribute value matches a given string pattern
IS NULL	Checks an attribute value is NULL
IN	Checks an attribute value matches any value within a value list
DISTINCT	Limits values to unique values

Sample Queries:

```
SELECT * FROM employees WHERE salary
between 10000 and 20000;

SELECT * FROM employees WHERE first_name
like 'Steven';

SELECT * FROM employees WHERE salary is
null;

SELECT * FROM employees where salary in
(10000,12000,20000);

SELECT DISTINCT(first_name) from
employees;
```

SQL Operators – Aggregations

Aggregation Functions	
Avg()	Returns the average value from specified columns
Count()	Returns number of table rows
Max()	Returns largest value among the records
Min()	Returns smallest value among the records
Sum()	Returns the sum of specified column values

Sample Queries:

```
SELECT avg(salary) FROM employees;

SELECT count(*)  FROM employees;

SELECT min(salary) FROM employees;

SELECT max(salary) FROM employees;

SELECT sum(salary) FROM employees;
```

SQL GROUP BY Clause

- Arrange identical data into groups.

e.g.,

```
SELECT max(salary),dept_id
FROM employees
GROUP BY dept_id
```

emp_id	first_name	last_name	salary	dept_id
103	Harry	Potter	20000	12
102	Edwin	Thomas	15000	11
101	Steven	Cohen	10000	10
100	Erik	John	10000	12

SQL HAVING Clause

- Used with aggregate functions due to its non-performance in the WHERE clause.
- Must follow the GROUP BY clause in a query and must also precede the ORDER BY clause if used.

employee_id	first_name	last_name	salary	dept_id
103	Harry	Potter	20000	12
102	Edwin	Thomas	15000	11
101	Steven	Cohen	10000	10
100	Erik	John	10000	12

e.g.,

```
SELECT AVG(salary),dept_id
FROM employees
GROUP BY dept_id
HAVING count(dept_id)>=2
```

SQL ORDER BY Clause

- Used to sort output of SELECT statement
- Default is to sort in ASC (Ascending)
- Can Sort in Reverse (Descending) Order with "DESC" after the column name

employee_id	first_name	last_name	salary
101	Steven	Cohen	10000
102	Edwin	Thomas	15000
103	Harry	Potter	20000

e.g.,

```
SELECT * FROM employees
ORDER BY salary DESC;
```

employee_id	first_name	last_name	salary
103	Harry	Potter	20000
102	Edwin	Thomas	15000
101	Steven	Cohen	10000

SQL UNION

- Used to combine the result-set of two or more SELECT statements removing duplicates
- Each SELECT statement within the UNION must have the same number of columns
- The selected columns must be of similar data types and must be in the same order in each SELECT statement
- More than two queries can be clubbed using more than one UNION statement

SQL UNION

Product1

CATEGORY_ID	PRODUCT_NAME
1	Nokia
2	Samsung
3	HP
6	Nikon

Product2

CATEGORY_ID	PRODUCT_NAME
1	Samsung
2	LG
3	HP
5	Dell
6	Apple
10	Playstation

e.g.,

```
SELECT product_name FROM product1
UNION
SELECT product_name FROM
product2;
```

PRODUCT_NAME
Nokia
Samsung
HP
Nikon
LG
Dell
Apple

SQL UNION ALL

- Used to combine the results of two SELECT statements including duplicate rows.
- The same rules that apply to the UNION clause will apply to the UNION ALL operator.

SYNTAX:

```
SELECT col1,col2... FROM table1
UNION ALL
SELECT col1,col2... FROM table2;
```

SQL UNION ALL

Product1

CATEGORY_ID	PRODUCT_NAME
1	Nokia
2	Samsung
3	HP
6	Nikon

Product2

CATEGORY_ID	PRODUCT_NAME
1	Samsung
2	LG
3	HP
5	Dell
6	Apple
10	Playstation

e.g.,

```
SELECT product_name FROM product1
UNION ALL
SELECT product_name FROM
product2;
```

PRODUCT_NAME
Nokia
Samsung
HP
Nikon
Samsung
LG
HP
Dell
Apple