

A close-up photograph of a person's hands at a desk. The left hand holds a black pencil, pointing it at a document. The right hand rests on the desk, wearing a gold-toned watch with a dark brown leather strap. The document is a business report titled "BUSINESS REPORT" and contains several charts: a pie chart, a bar chart with five bars, a line graph with two lines, and a large blue and yellow area chart. The background is blurred, showing a typical office environment.

SQL(Week 1)

DSBA CURRICULUM DESIGN

FOUNDATIONS

**Python for Data
Science**

**Statistical Methods
for Decision Making**

CORE COURSES

Advanced Statistics

Data Mining

Predictive Modelling

Machine Learning

Data Visualization

SQL (Week-1/3)

**Time Series
Forecasting**

DOMAIN APPLICATIONS

**Financial Risk
Analytics**

**Marketing Retail
Analytics**

Table of Contents

- Introduction to SQL
- Subsets of SQL
 - DDL
 - DML
 - DCL
 - TCL
- SQL Operators
- SQL Functions
- SQL SET Operators

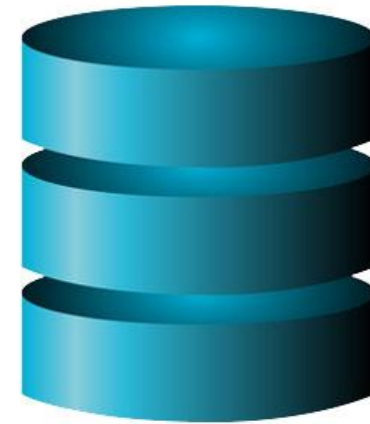
Introduction to Database

What is Database?

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

Examples:

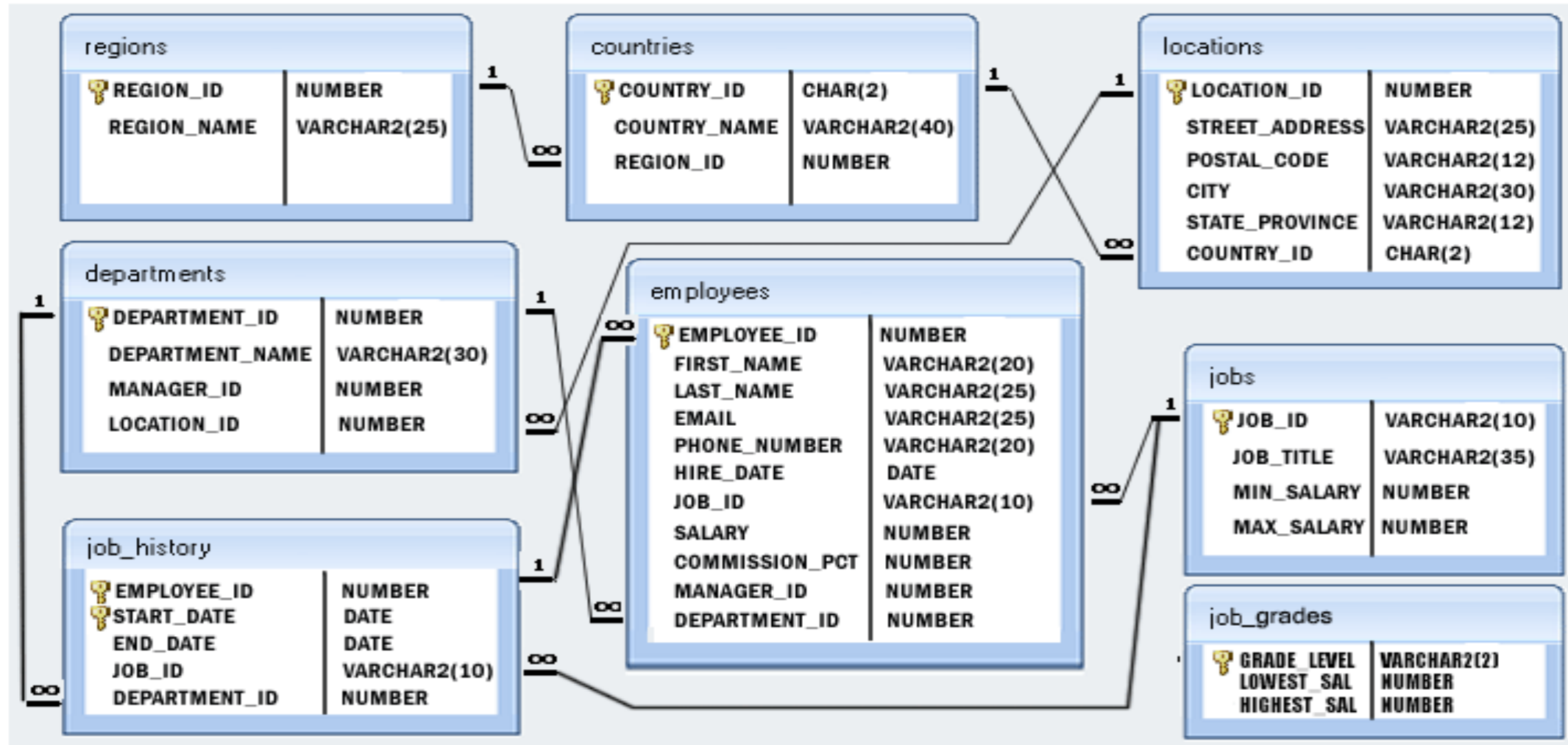
- Telephone directory
- Customer data
- Product inventory
- Visitors' register
- Weather records



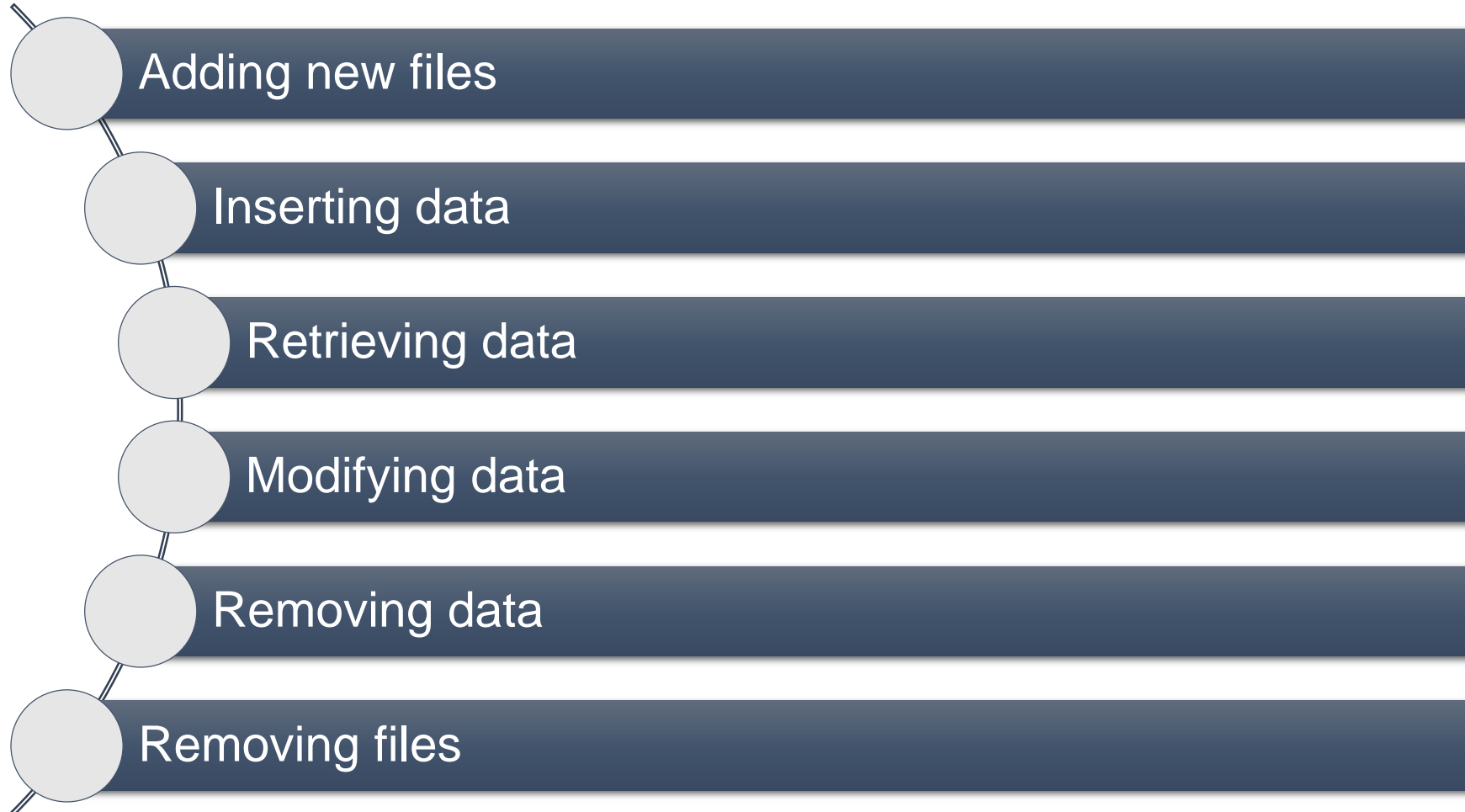
Types of Data Models

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

E/R Diagram



DBMS Operations



Advantages of DBMS

- Sharing of data across applications
- Reduced data redundancy
- Enhanced security mechanism
- Data independence
- Better flexibility
- Enforce integrity constraints
- Better transaction support
- Enforce standards
- 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



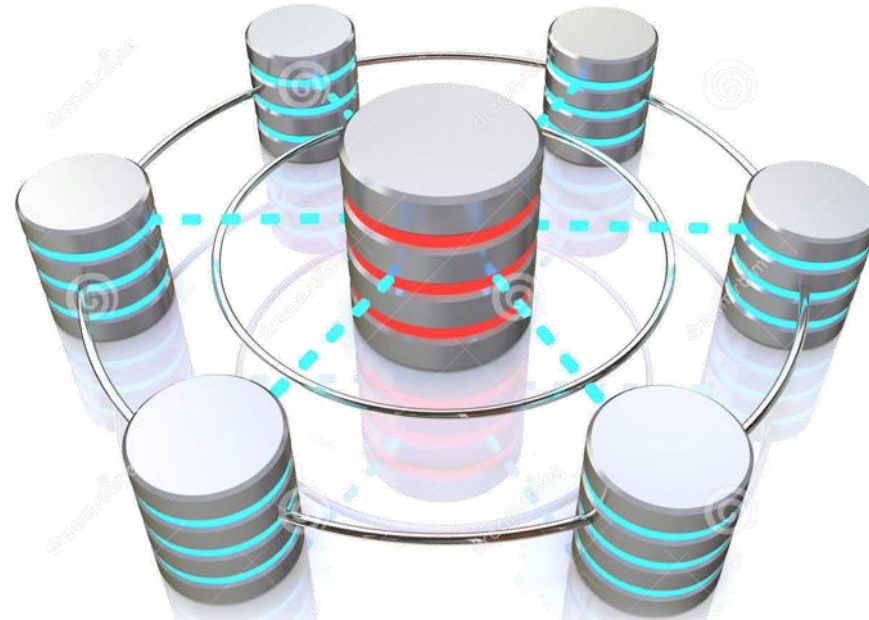
RDBMS VS TRADITIONAL APPROACH

- 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 or a navigational form. This means that a single data unit will have one parent node and zero, one or more children nodes.

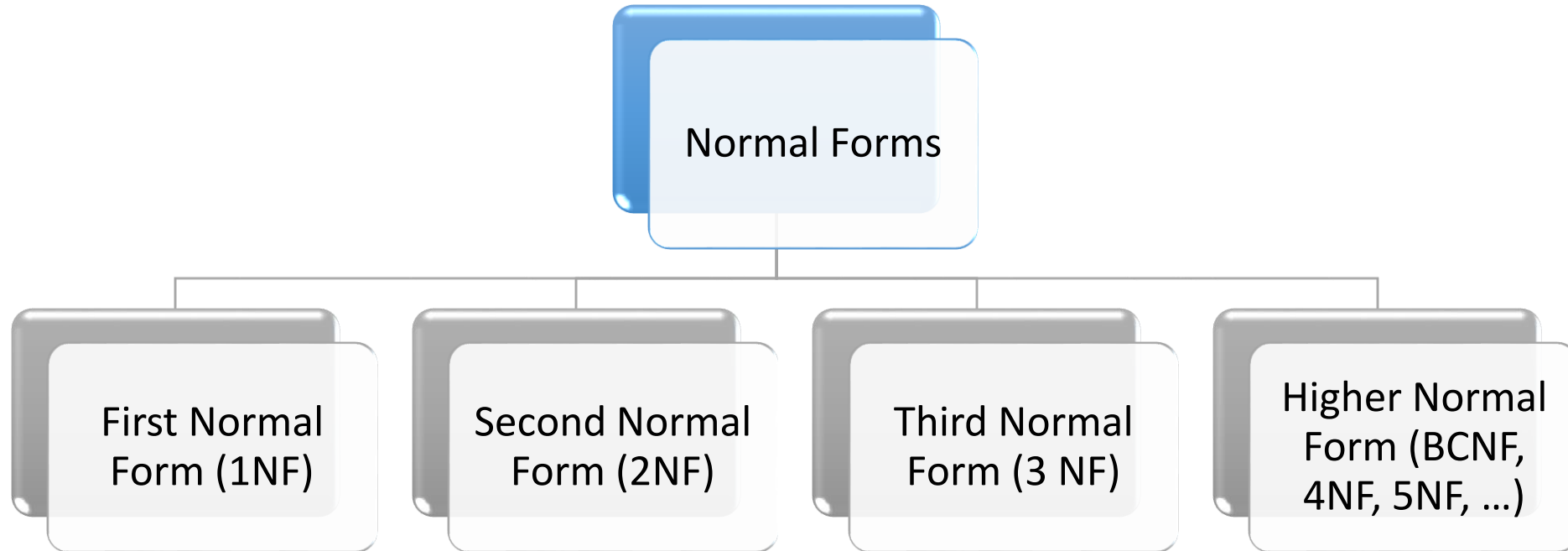
Normalization

Normalization

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

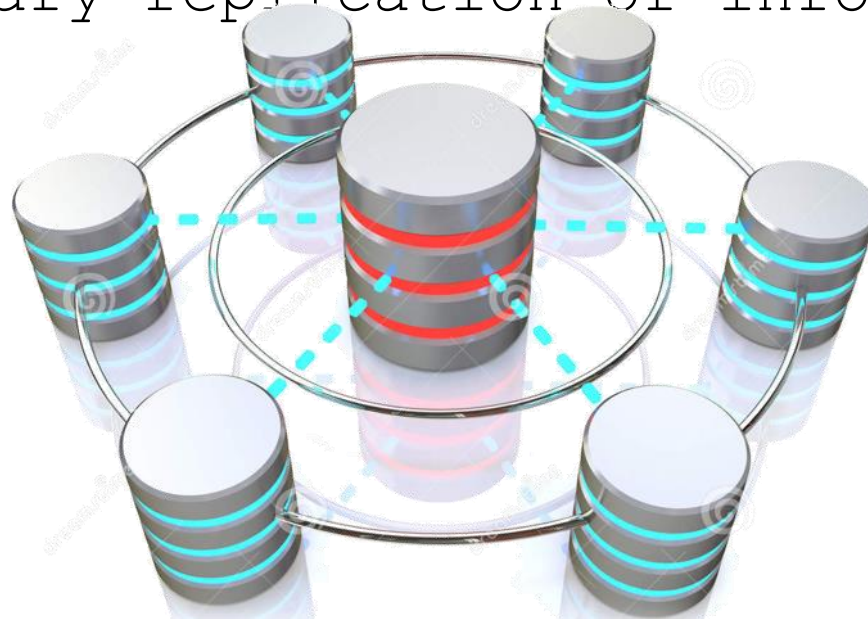


Normalization and Normal Forms



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

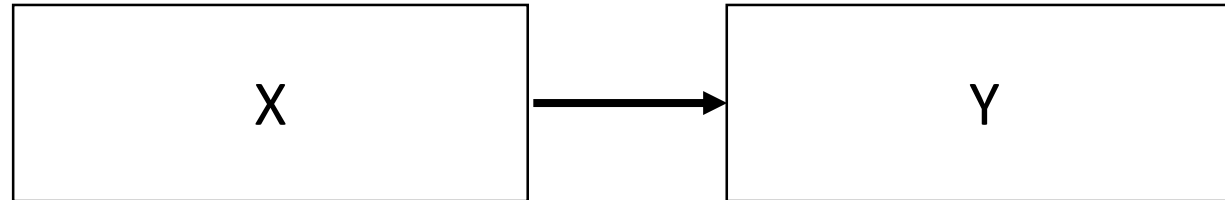
Student_Details	Course_details	Pre-requisite	Result_details
0101 Tim 11/4/1985	M1 Advance Maths 7	Basic Math	02/11/2015 82 A
0102 Rob 10/04/1986	P4 Advance Physics 8	Basic Physics	21/11/2015 89 A
0103 Mary 11/07/1985	B3 Advance Biology 10	Basic Biology	12/11/2015 62 B
0104 Rob 10/04/1986	H6 Advance History 9	Basic History	21/11/2015 89 A
0105 Tom 03/08/1988	C3 Advance Chemistry 11	Basic Biology	12/11/2015 50 C

Functional Dependency

- Consider the relation
 - Result (Student#, Course#, Course Name#, 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# - Course Name
 - Marks# - Grade

Functional Dependency

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



Functional Dependency Types

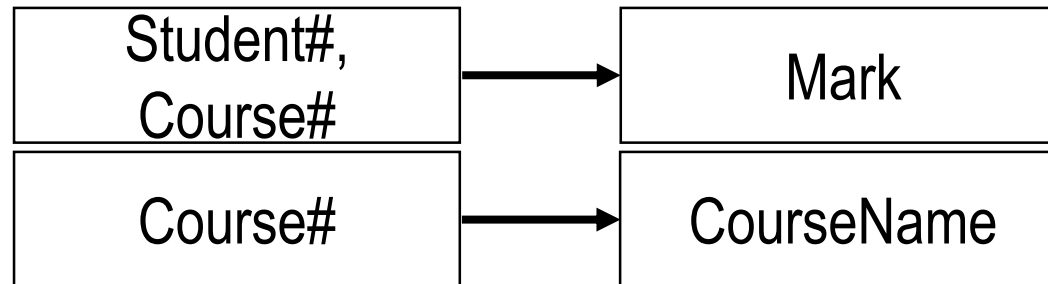
Partial Functional Dependency

Transitive Dependency

Functional Dependency Types

Partial Functional Dependency

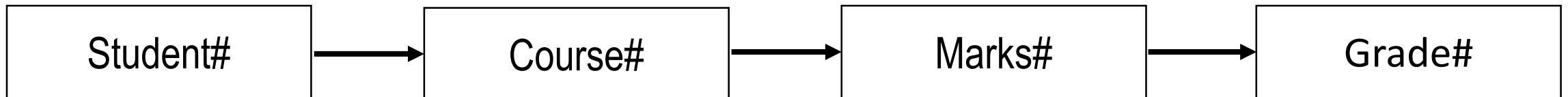
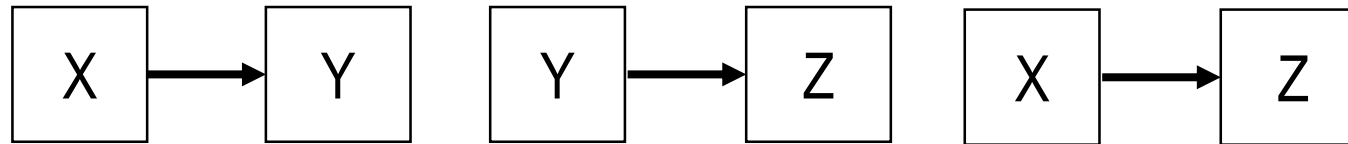
- Attribute Y is partially dependent on attribute X, if and only if it is dependent on the subset of attribute X.
- REPORT (Student#, Course#, StudentName, CourseName, Marks, Grade)



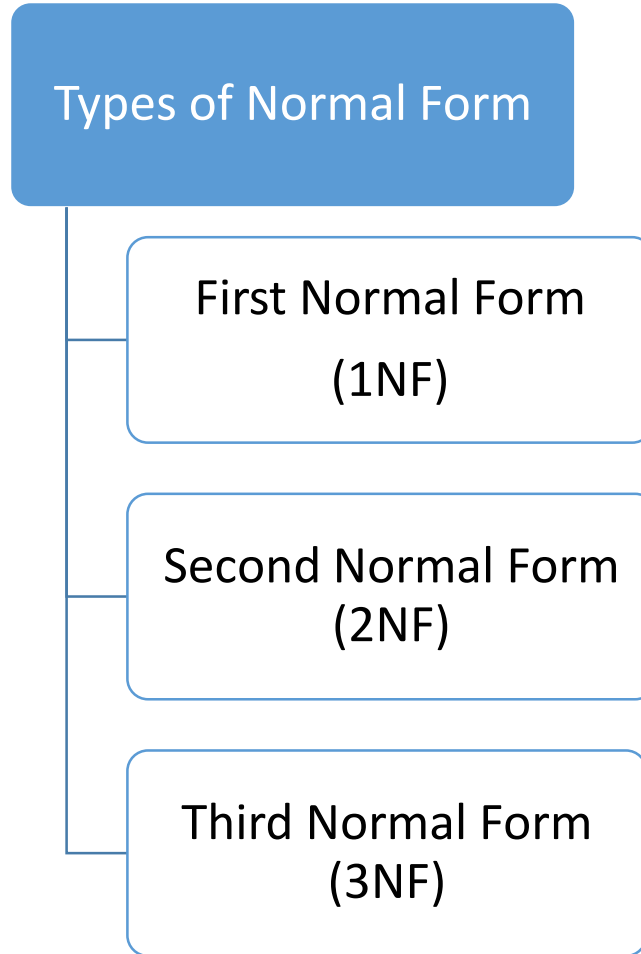
Functional Dependency Types

Transitive Dependency

X, Y, Z are three attributes



Normalization



Second Normal Form – (2NF)

Student Marks Table in 1NF

Student#	Student_Name	DOB	Course #	CourseName	Pre Requisite	Duration in days	Date of Exam	Marks	Grade
0101	Tim	11/4/1985	M1	Advance Math	Basic Math	7	02/11/2015	82	A
0102	Rob	10/04/1986	P4	Advance Physics	Basic Physics	8	21/11/2015	89	A
0103	Mary	11/07/1985	B3	Advance Biology	Basic Biology	10	12/11/2015	62	B

Second Normal Form – (2NF)

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

Partial
Dependenc
y with the
Key
attribute

Split/Decompose the
tables to remove partial
dependencies

Second Normal Form – (2NF)

Student Table

<u>Student#</u>	Student_Name	Date Of Birth
0101	Tim	11/4/1985
0102	Rob	10/04/1986
0103	Mary	11/07/1985

Result Table

<u>Student#</u>	<u>Course#</u>	Marks	Grade
0101	M1	82	A
0102	P4	89	A
0103	B3	62	B

Course Table

<u>Course#</u>	CourseName	Prerequisite	Durationindays	Date Of Exam
M1	Advance Math	Basic Math	7	02/11/2015
P4	Advance Physics	Basic Physics	8	21/11/2015
B3	Advance Biology	Basic Biology	10	12/11/2015

Third Normal Form – (3NF)

Types of Normal Form

First Normal Form
(1NF)

Second Normal Form
(2NF)

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	Grade
0101	M1	82	A
0102	P4	89	A
0103	B3	62	B

Student# ,Course# → Marks

Student#, Course# → Grade

Marks → Grade

Student#,Course#→ Marks→ Grade:TD



Remove

Third Normalization – (3NF)

Result Table

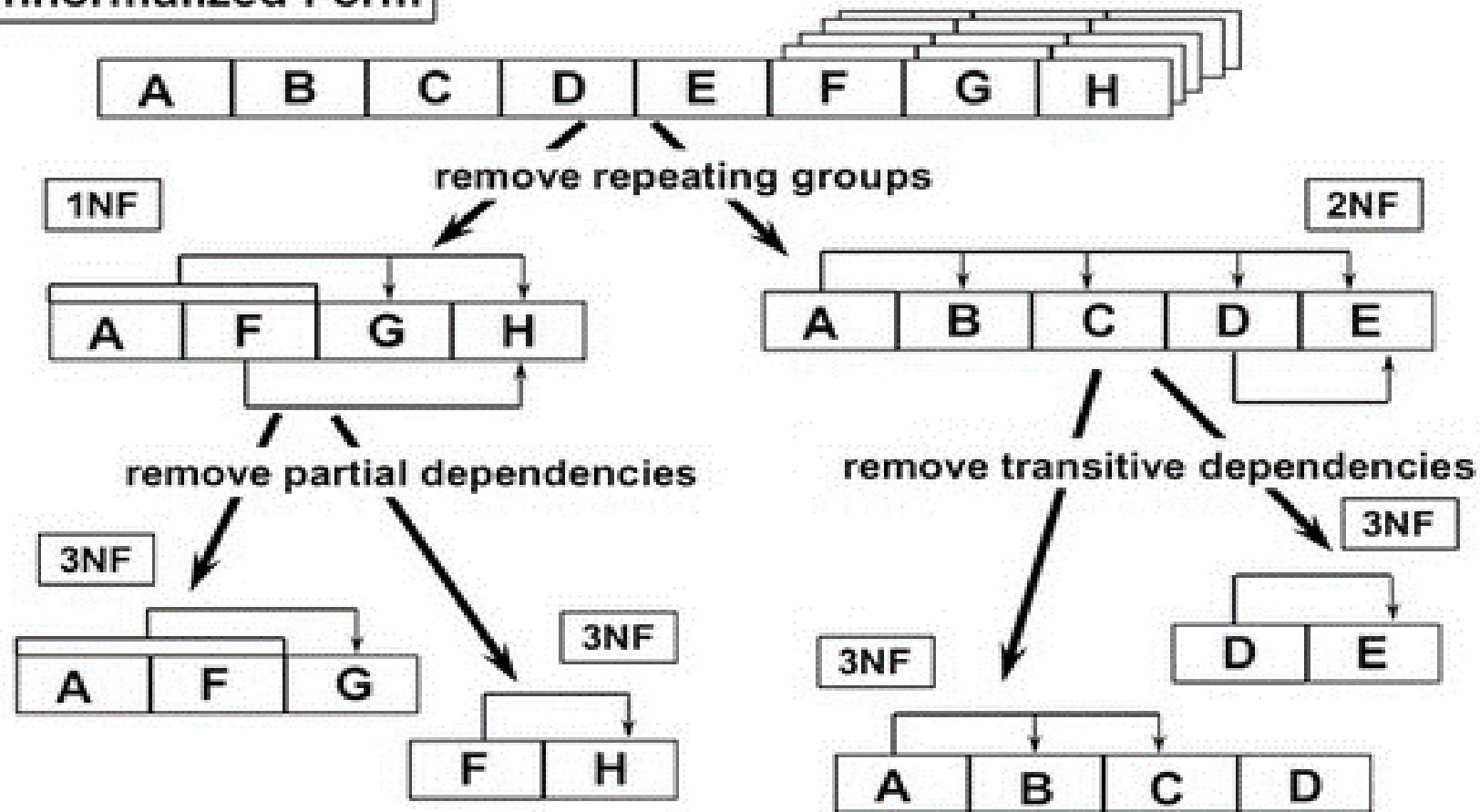
<u>Student#</u>	<u>Course#</u>	Marks
0101	M1	82
0102	P4	89
0103	B3	62

Marks Grade Table

Marks	Grade
82	A
89	A
62	B

Normalization In Nutshell

Unnormalized Form



Advantages And Disadvantages Of Normalization

ADVANTAGES	DISADVANTAGE
<ul style="list-style-type: none">• Based on mathematical foundation• Removes the redundancy to a large extent• After 3NF, data redundancy is minimized to the extent of foreign keys• Removes the anomalies present in INSERTs, UPDATEs and DELETEs	<ul style="list-style-type: none">• Data retrieval or SELECT operation performance will be severely affected• Normalization might not always represent real world scenarios

Introduction to SQL

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 **access data** in the relational database management systems.
- Allows users to **describe the data**.
- Allows users to **define the data** in a database and manipulate that data.

SQL Terms

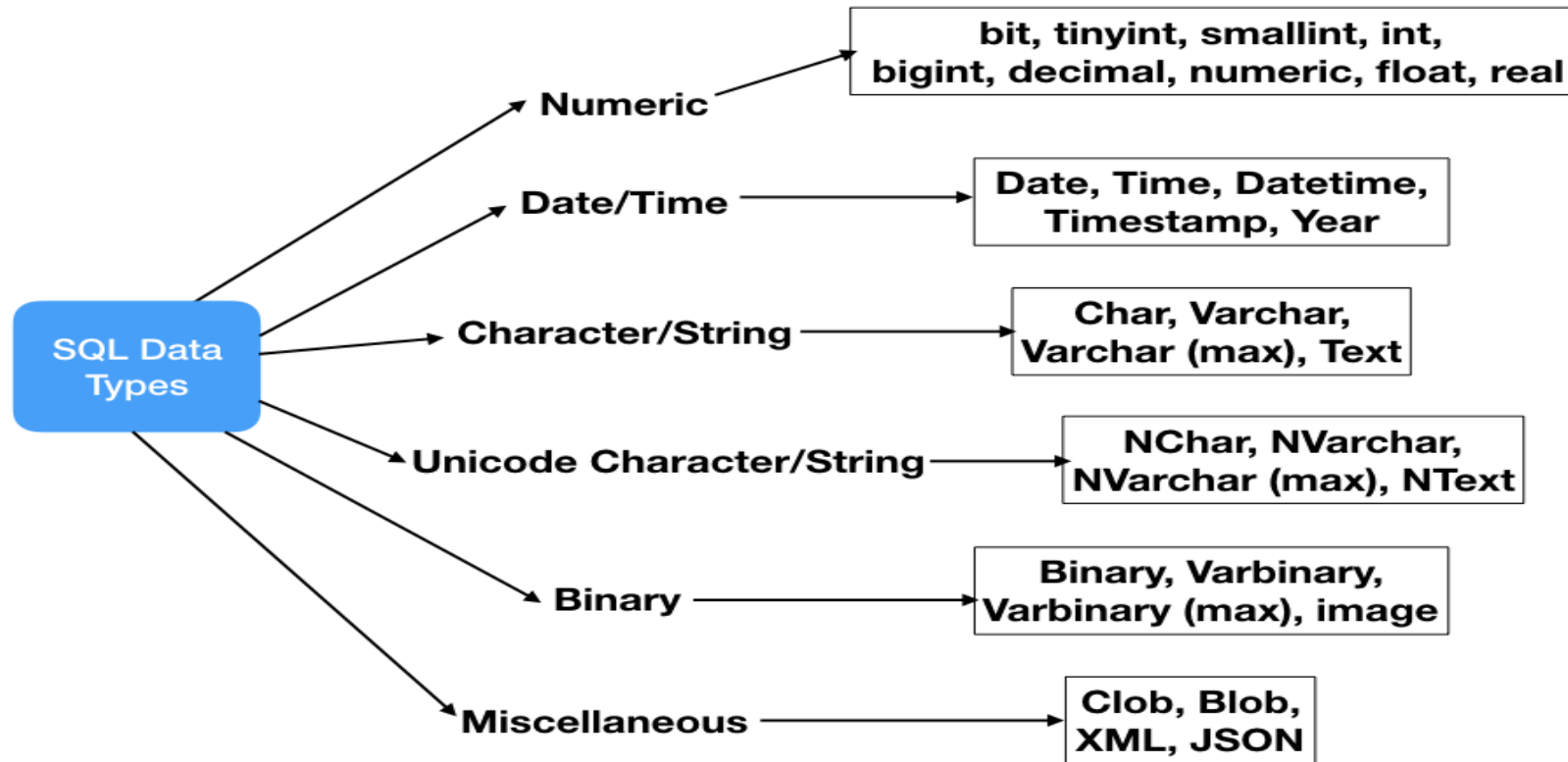
Data

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

Database

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

SQL Data Types



SQL Constraints

- Constraints are the rules enforced on data columns on a table.
- These are used to limit the type of data that can go into a table.

Constraint	Description
NOT NULL	Ensures that a column cannot 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	Uniquely identifies each row/record in a database table
FOREIGN	Uniquely identifies a row/record in any another database table
CHECK	The CHECK constraint ensures that all values in a column satisfy certain conditions.
INDEX	Used to create and retrieve 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	Create objects in the database
ALTER	Alters the structure of the database object
DROP	Delete objects from the database
TRUNCATE	Remove all records from a table permanently
COMMENT	Add comments to the data dictionary
RENAME	Rename an object

DDL - Data Definition Language - Create Command

```
CREATE TABLE employees (  
    employee_id INT (11) UNSIGNED NOT NULL,  
    first_name VARCHAR(20),  
    last_name VARCHAR(25) NOT NULL,  
    salary int(7) NOT NULL,  
    PRIMARY KEY (employee_id));
```

employee_id	first_name	last_name	salary

DDL - Data Definition Language - Alter Command

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

employee_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;
```

employee_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;
```

employee_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;
```

employee_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 unwanted/all records from a table

DML – Data Manipulation Language – INSERT Command

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

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

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

employee_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 employee_id=101;
```

employee_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 employee_id  
IN (101,103);
```

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

DCI – Data Control Language

Command	Description
GRANT	Gives user's access privileges to database
REVOKE	Withdraw access privileges given with the GRANT command

TCL - Transaction Control

Command	Description
COMMIT	Save work done
ROLLBACK	Restore database to original state since the last COMMIT
SAVEPOINT	Identify a point in a transaction to which you can later roll back

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.,

e.g.

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

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

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

employee_id	first_name	last_name	salary
101	Steven	Cohen	10000

SQL Operators – Logical

OPERATOR	ILLUSTRATIVE EXAMPLE	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 Operator's	
SYMBOL	MEANING
=	Equal to
<	Less than
<=	Less than or equal to
>	Greater than
>=	Greater 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;
```

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

SQL Operators – Special

Special Operator's	
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 Functions

SQL Operators – Aggregations

Aggregation function's	
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.
- This GROUP BY clause follows the WHERE clause in a SELECT statement and precedes the ORDER BY clause if used.

e.g.,

```
SELECT  
SUM(salary), department_id  
FROM employees  
WHERE salary >=15000  
GROUP BY department_id
```

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

SUM(salary)	department_id
20000	12
15000	11

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.

e.g.,

```
SELECT AVG(salary), department_id
FROM employees
WHERE salary >=10000
GROUP BY department_id
HAVING count(department_id) >=2
```

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

AVG(salary)	department_id
15000	12

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

e.g.,

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

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

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

SQL Set Operators

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 a.col1,b.col2,...,a.coln FROM table1 a,table1 b WHERE a.commonfield = b.commonfield
```

```
UNION ALL
```

```
SELECT a.col1, b.col2,..., a.coln FROM table1 a, table1 b
```

```
WHERE a.commonfield = b.commonfield
```

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
Playstation

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
Playstation

SQL Case

SQL CASE Statement

- The CASE statement goes through conditions and returns a value when the first condition is met (like an IF-THEN-ELSE statement).
- So, once a condition is true, it will stop reading and return the result. If no conditions are true, it returns the value in the ELSE clause.
- If there is no ELSE part and no conditions are true, it returns NULL.

```
CASE
  WHEN condition1 THEN result1
  WHEN condition2 THEN result2
  WHEN conditionN THEN resultN
  ELSE result
END;
```

```
SELECT OrderID, Quantity,
CASE
  WHEN Quantity > 30 THEN 'The quantity is
greater than 30'
  WHEN Quantity = 30 THEN 'The quantity is
30'
  ELSE 'The quantity is under 30'
END AS QuantityText
FROM OrderDetails;
```

```
CASE department_name
WHEN 'CS'
THEN UPDATE Faculty SET department='Computer
Science';
WHEN 'EC'
THEN UPDATE Faculty SET department='Electronics
and Communication';
ELSE UPDATE Faculty SET department='Humanities
and Social Sciences';
END CASE
```




Data Science @ Work

Apply **Data Science at your workplace** to gain some instant benefits:

- Get noticed by your management with your outstanding analysis backed by data science.
- Create an impact in your organization by taking up small projects/initiatives to solve critical issues using data science.
- Network with members from the data science vertical of your organization and seek opportunities to contribute in small projects.
- Share your success stories with us and the world to position yourself as a subject matter expert in data science.



ANY QUESTIONS



HAPPY LEARNING

Excelerate CareerPrep:

Career services by Great Learning.

The CareerPrep is an organized way to learn about all of the topics asked in interviews for Data Science jobs.

- Complete Excelerate career prep and learn how to crack an interview with ease.
- Understand the different kind of roles available in the data science ecosystem along with their skill set requirements
- Build a resume that has a higher probability of getting shortlisted by recruiters
- Prepare for different data science roles by reviewing 400+ questions asked in actual interviews
- Understand and implement frameworks to solve guesstimates, logical and aptitude problems often used in interviews
- Learn from the experience of alumni who have successfully transitioned
- Speed up your path towards a transition to a Data science role.

Complete Career Prep today to speed up your path towards DS transition. You'll have a 40% better chance of getting hired than other candidates.

Note: CareerPrep is mandatory step in unlocking the job board.