Lesson 6

Normalization

Dr. Bright Gee Varghese

CS-401 MODERN PROGRAMMING PRACTICES

	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
OOAD / RDBMS / OOP	**AM: Lesson 1:** *The OO Paradigm for Building Software Solutions* **PM: Lab 1**	**AM: Lesson 2:** *Associations among Objects and Classes* [Lab 1 due 10 AM] **PM: Lab 1 solutions, Lab 2**	**AM: Lesson 3:** *Inheritance and Composition* [Lab 2 due 10 AM] **PM: Lab 2 solutions, Lab 3**	**AM: Lesson 4**: *Interaction Diagrams* [Lab 3 due 10 AM] **PM: Lab 3 solutions, Lab 4**	**AM: Lesson 5**: *Inheritance and Abstraction* [Lab 4 due 10 AM] **PM: Lab 4 solutions, Lab 5**	**AM: Lesson 6:** *Relational Model, View & Normalization* [Lab 5 due 10 AM] **Lab 5 solutions**
	AM: Lesson 7: *SQL (DML&DDL)* **PM: Lab 6, 7**	**AM: Lesson 8**: *Index, SQL Injection, JDBC application & Intro to Maven* [Lab 6 due 10 AM] [Lab 7 due 5 PM] **PM: Lab 6 solutions, Lab 8**	**AM:** Review for Midterm exam **Lab 7 & 8 solutions** **PM:** *Study for Midterm*	**MIDTERM EXAM**	**AM: Lesson 9**: *Interfaces in Java 8 and the Object Superclass* **PM: Lab 9**	**AM: Lesson 10**: *Functional Programming in Java* [Lab 9 due 10 AM] **Lab 9 solutions**
OOP	**AM: Lesson 11**: ***The Stream API*** **PM: Lab 10**	**AM: Lesson 11**: ***The Stream API*** [Lab 10 due 10 AM] **PM: Lab 10 solutions, Lab**	**AM: Lesson 12** *Best Programming Practices with Java 8* [Lab 11 due 10 AM] **PM: Lab 11 solutions, Lab**	**AM: Lesson 13** *Generic Programming* **PM: Lab 12 solutions, Lab 13**	**AM**: Review for Final exam **Lab 13 solutions**	**Final Exam**
	AM: Java Swing **PM: Course Project**	**Course Project**	**Course Project**	**AM:** *Project Presentation – Connect whole with parts*		

Wholeness of the Lesson

The relational model is an application of elementary relation theory to systems that provide shared access to large banks of formatted data.

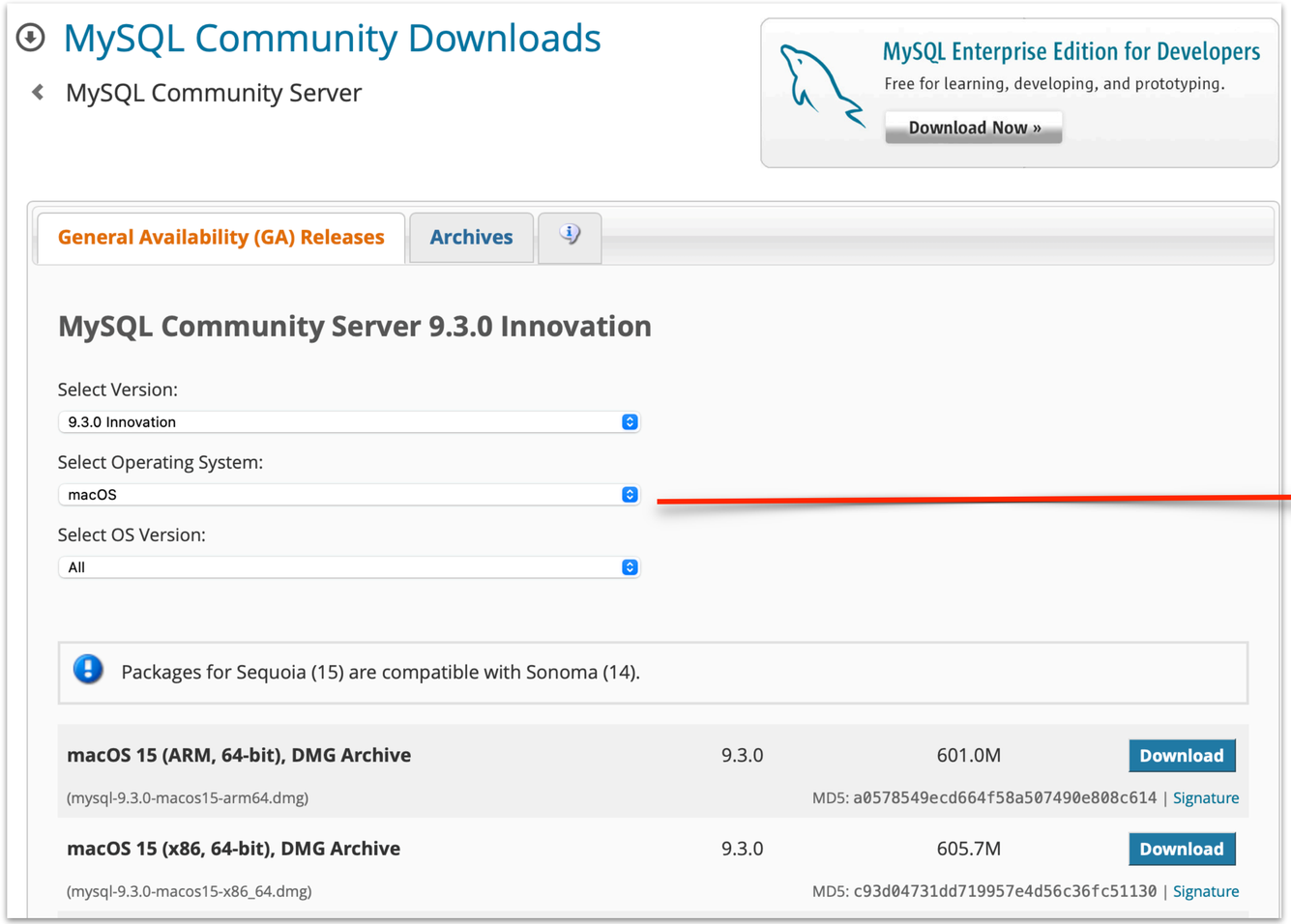
Science & Technology of Consciousness: Consciousness-based education is an application of Maharishi's Vedic Science in the field of education.

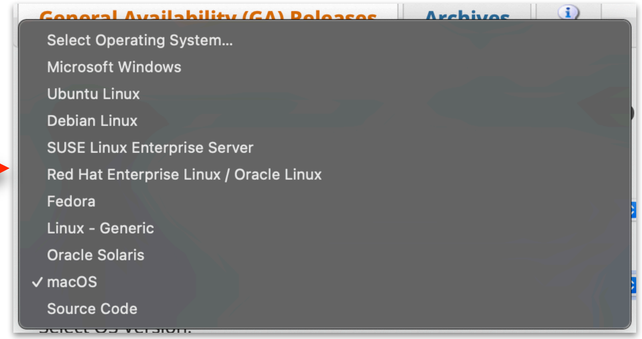
Lesson Outline

- Relational Model
- Introduction to Normalization
- Functional Dependency
- The process of normalization

Install MySQL

https://dev.mysql.com/downloads/mysql/





Connect MySQL

Using terminal

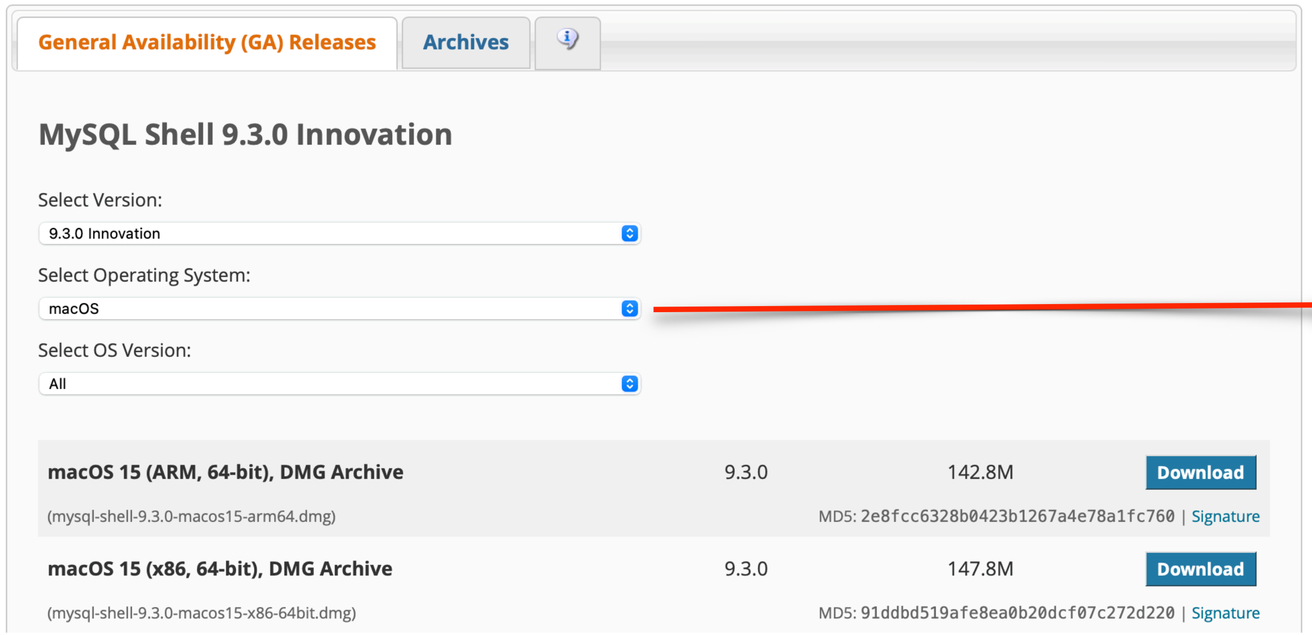
```
bright~$mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 13
Server version: 9.3.0 MySQL Community Server - GPL

Copyright (c) 2000, 2025, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

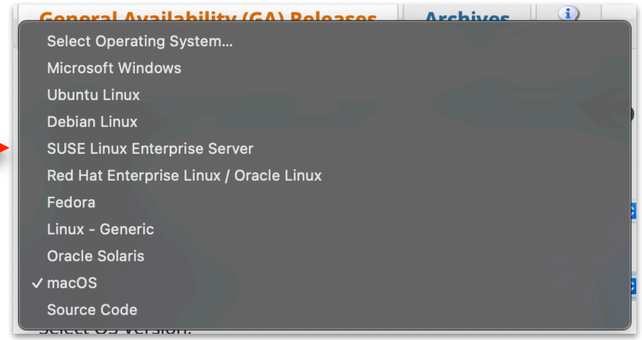
Install MySQL Shell

https://dev.mysql.com/downloads/shell/

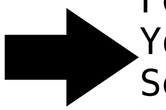
MySQL Community Downloads

MySQL Shell





MySQL Shell



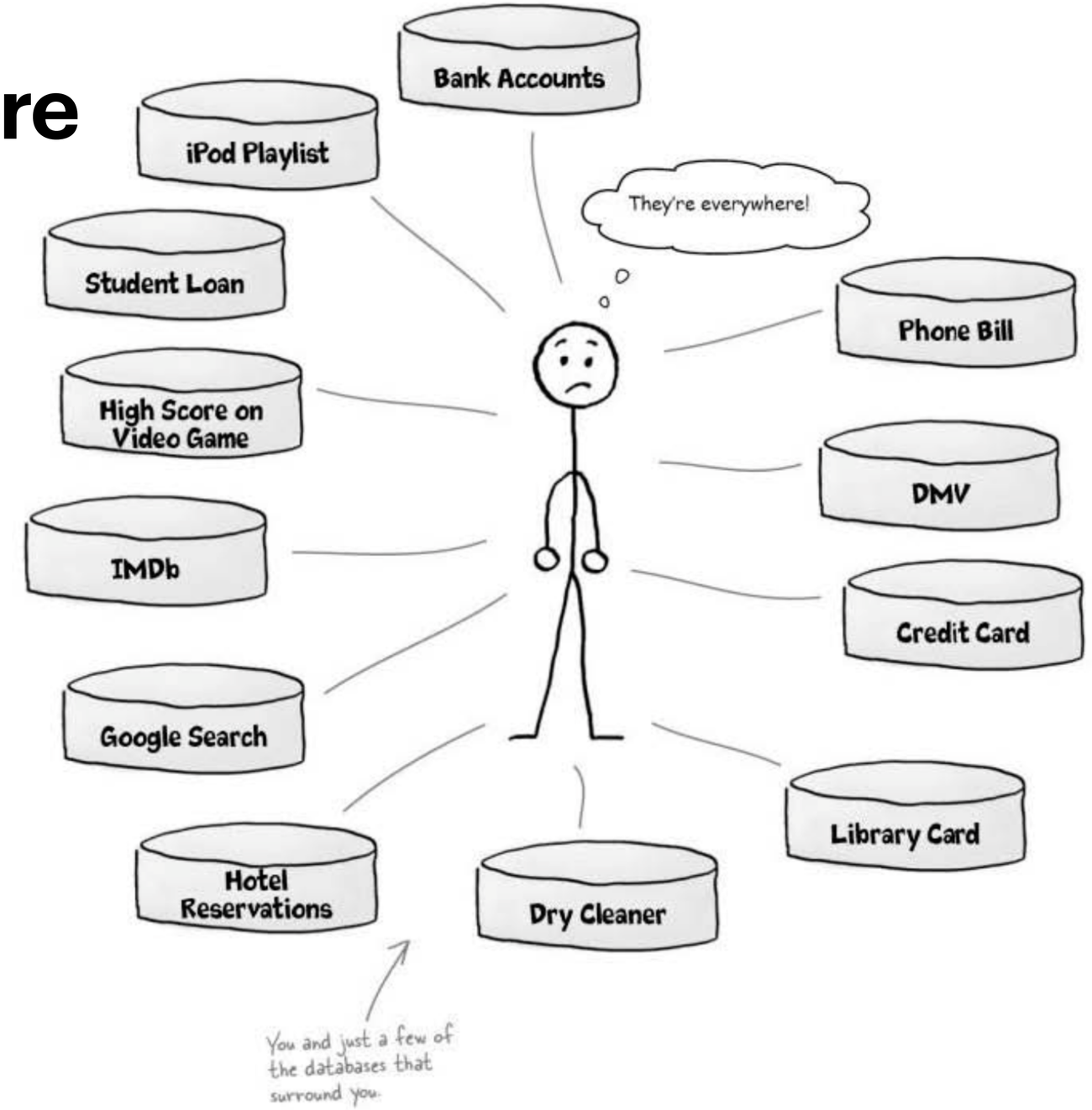
```
bright~$mysqlsh root@localhost
MySQL Shell 9.3.0

Copyright (c) 2016, 2025, Oracle and/or its affiliates.
Oracle is a registered trademark of Oracle Corporation and/or its affiliates.
Other names may be trademarks of their respective owners.

Type '\help' or '\?' for help; '\quit' to exit.
Creating a session to 'root@localhost'
Fetching global names for auto-completion... Press ^C to stop.
Your MySQL connection id is 21
Server version: 9.3.0 MySQL Community Server - GPL
No default schema selected; type \use <schema> to set one.
 MySQL  localhost:3306 ssl  SQL  \py
Switching to Python mode...
 MySQL  localhost:3306 ssl  Py  \js
Switching to JavaScript mode...
 MySQL  localhost:3306 ssl  JS  \sql
Switching to SQL mode... Commands end with ;
 MySQL  localhost:3306 ssl  SQL
```

What is Database?

- A database is a container that holds tables and other SQL structures related to those tables.
  - Other SQL structures
    - Views
    - Indexes
    - Stored Procedures
    - ...

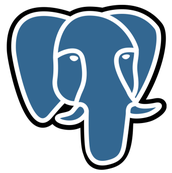
In diagrams and flow charts, databases are depicted as cylinders. So when you see this, think database.

Database Everywhere



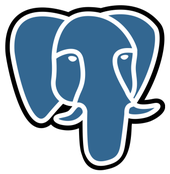
What is DBMS?

- A Database Management System (DBMS) is software that is used to store, manage, and retrieve data efficiently in a structured way. It acts as an interface between the database and the users or applications that access it.




Functions of DBMS

- Data Storage & Retrieval – Stores and retrieves data efficiently.
- Data Security & Integrity – Ensures only authorized users can access data.
- Concurrency Control – Manages multiple users accessing data simultaneously.
- Backup & Recovery – Protects data from crashes and failures.
- Query Processing – Allows users to retrieve data using SQL queries.




Relational Model

- The Relational Model is a conceptual framework for organizing and managing data in a database.
- It was introduced by E.F. Codd in 1970 and is the foundation for modern relational database management systems (RDBMS).

Ref:

https://www.seas.upenn.edu/~zives/03f/cis550/codd.pdf

Key Concepts of Relational Model

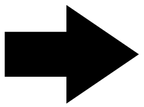
ID	NAME	CLASS	MARK	GENDER
1	John Deo	Four	75	female
2	Max Ruin	Three	85	male
3	Arnold	Three	55	male
4	Krish Star	Four	60	female
5	John Mike	Four	60	female
6	Alex John	Four	55	male
7	My John Rob	Five	78	male
8	Asruid	Five	85	male
9	Tes Qry	Six	78	male
10	Big John	Four	55	female

- Tables (Relations):
  - Data is organized into tables, also called relations.
  - Each table represents an entity or a relationship between entities.
    - Example: A Students table represents the entity "Student."
- Rows (Tuples):
  - Each row in a table represents a single record or instance of an entity.
    - Example: A row in the Students table represents a specific student.
- Columns (Attributes):
  - Each column represents an attribute or property of the entity.
    - Example: Columns in the Students table might include ID, Name, Class, Mark and Gender.
- Constraints:
  - Rules enforced on data to maintain integrity.
    - Examples: Primary Key, Foreign Key, Unique, Not Null, Check.

Create a database

CREATE DATABASE db_name;

```
 MySQL  localhost:3306 ssl  SQL  CREATE DATABASE ecommerce_db;
Query OK, 1 row affected (0.0038 sec)
 MySQL  localhost:3306 ssl  SQL  show databases;
+--------------------+
| Database           |
+--------------------+
| ecommerce_db       |
| information_schema |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
5 rows in set (0.0009 sec)
 MySQL  localhost:3306 ssl  SQL
```



Switch to a particular db

use db_name;

```
 MySQL  localhost:3306 ssl  SQL  use ecommerce_db;
Default schema set to `ecommerce_db`.
Fetching global names, object names from `ecommerce_db` for auto-completion... Press ^C to
stop.
 MySQL  localhost:3306 ssl  ecommerce_db  SQL
```

This means that all your SQL queries will run against the ecommerce_db database unless you specify another database.

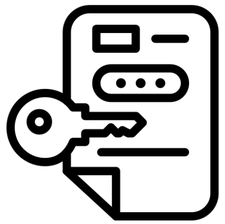
Introduction to Database Keys

What are Database Keys?

- Special attributes used to identify records uniquely
- Help establish relationships between tables
- Enforce data integrity and constraints
- Essential for effective database design

Types of Keys
Relational Model

- Keys are one of the basic requirements of a relational database model.
- It is widely used to identify the tuples(rows) uniquely in the table.
  - Candidate Key
  - Primary Key
  - Foreign Key
  - Composite Key
  - Natural Key
  - Surrogate Key



Candidate Keys

Candidate Key
Primary Key
Foreign Key
Composite Key
Natural Key
Surrogate Key

A Candidate Key is a set of one or more fields/columns that can identify a record uniquely in a table.

Examples of candidate keys

The candidate keys Roll No., Stud ID, and Email in the table enable us to identify each student record uniquely.

StudID	Roll No	First Name	Last Name	Email
1	11	Tom	Prince	abc@gmail.com
2	12	Nick	Wright	xyz@gmail.com
3	13	Dana	Natan	mno@gmail.com

Candidate Keys

There can be multiple Candidate Keys in one table.

Each Candidate Key can work as a Primary Key.

The value for the Candidate key is always unique and non-null for all the tuples types.

One thing to be remembered is that every table has to have at least one Candidate key, but there can be more than one candidate key can be there in a table.

StudID	Roll No	First Name	Last Name	Email
1	11	Tom	Prince	abc@gmail.com
2	12	Nick	Wright	xyz@gmail.com
3	13	Dana	Natan	mno@gmail.com

Primary Key

The Unique Identifier

- Candidate Key
- Primary Key
- Foreign Key
- Composite Key
- Natural Key
- Surrogate Key

- The PRIMARY KEY constraint uniquely identifies each record in a table.
- Primary keys must contain UNIQUE values, and cannot contain NULL values.
- A table can have only ONE primary key; and in the table, this primary key can consist of single or multiple columns (fields).

product_id	product_name	price
1	Laptop	1200.50
2	Wireless Mouse	25.99
3	External Hard Drive (1TB)	79.95
4	4K Monitor	350.00
5	Mechanical Keyboard	129.75

Primary Key
Example

Syntax:

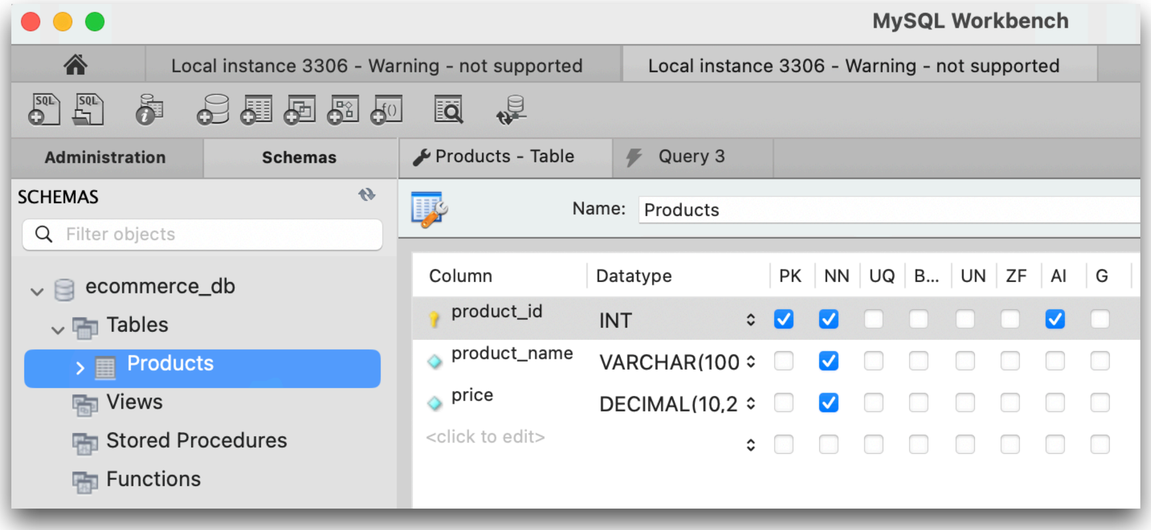
```
CREATE TABLE table_name (
    column1 datatype,
    column2 datatype,
    column3 datatype,
   ....
);
```

```
+------------+---------------------------+---------+
| product_id | product_name              | price   |
+------------+---------------------------+---------+
|
+------------+---------------------------+---------+
```

```
MySQL  localhost:3306 ssl  ecommerce_db  SQL  CREATE TABLE Product (
                                                 product_id INT AUTO_INCREMENT,
                                                 product_name VARCHAR(100) NOT NULL,
                                                 price DECIMAL(10, 2) NOT NULL,
                                                 PRIMARY KEY (product_id)
                                             );
Query OK, 0 rows affected (0.0094 sec)
MySQL  localhost:3306 ssl  ecommerce_db  SQL
```

Primary Key
Example

```
MySQL  localhost:3306 ssl  ecommerce_db  SQL
CREATE TABLE Product (
    product_id INT AUTO_INCREMENT,
    product_name VARCHAR(100) NOT NULL,
    price DECIMAL(10, 2) NOT NULL,
    PRIMARY KEY (product_id)
);
Query OK, 0 rows affected (0.0094 sec)
MySQL  localhost:3306 ssl  ecommerce_db  SQL
```



Syntax:

```
CREATE TABLE table_name (
    column1 datatype,
    column2 datatype,
    column3 datatype,
   ....
);
```

Primary Key

Example

Syntax:

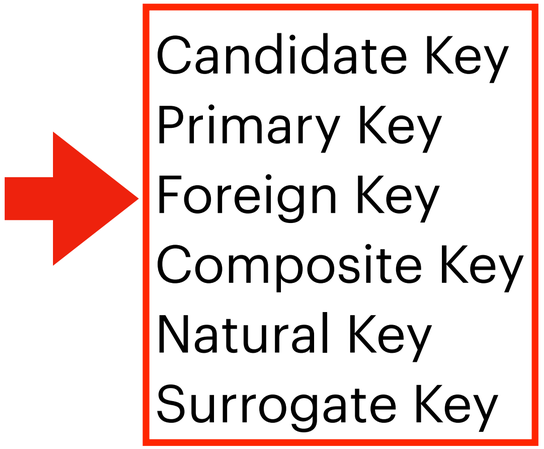
```
INSERT INTO table_name (column1, column2, column3, ...)
VALUES (value1, value2, value3, ...);
```

```
MySQL  localhost:3306 ssl  ecommerce_db  SQL
INSERT INTO Product(product_name, price) VALUES
('Laptop', 1200.50),
('Wireless Mouse', 25.99),
('External Hard Drive (1TB)', 79.95),
('4K Monitor', 350.00),
('Mechanical Keyboard', 129.75);
Query OK, 5 rows affected (0.0184 sec)

Records: 5  Duplicates: 0  Warnings: 0
MySQL  localhost:3306 ssl  ecommerce_db  SQL
```

```
MySQL  localhost:3306 ssl  ecommerce_db  SQL  select * from Product;
+------------+---------------------------+---------+
| product_id | product_name              | price   |
+------------+---------------------------+---------+
|          1 | Laptop                    | 1200.50 |
|          2 | Wireless Mouse            |   25.99 |
|          3 | External Hard Drive (1TB) |   79.95 |
|          4 | 4K Monitor                |  350.00 |
|          5 | Mechanical Keyboard       |  129.75 |
+------------+---------------------------+---------+
5 rows in set (0.0004 sec)
MySQL  localhost:3306 ssl  ecommerce_db  SQL
```

Foreign Key

Establishing Relationships



- The FOREIGN KEY constraint is used to prevent actions that would destroy links between tables.
- A FOREIGN KEY is a field (or collection of fields) in one table, that refers to the PRIMARY KEY in another table.
- The table with the foreign key is called the child table, and the table with the primary key is called the referenced or parent table.

```
MySQL  localhost:3306 ssl  ecommerce_db  SQL  select * from Customer;
+-------------+------------+-----------+---------------------------+-------------------------------+--------------+
| customer_id | first_name | last_name | email                     | address                       | phone_number |
+-------------+------------+-----------+---------------------------+-------------------------------+--------------+
|           1 | Alice      | Smith     | alice.smith@example.com   | 123 Main St, Anytown, CA      | 555-1234     |
|           2 | Bob        | Johnson   | bob.johnson@example.com   | 456 Oak Ave, Someville, NY    | 555-5678     |
|           3 | Charlie    | Brown     | charlie.brown@example.com | 789 Pine Ln, Othercity, TX    | 555-9012     |
|           4 | Diana      | Garcia    | diana.garcia@example.com  | 101 Elm Rd, Anytown, CA       | 555-3456     |
|           5 | Eve        | Williams  | eve.williams@example.com  | 222 Maple Dr, Someville, NY   | 555-7890     |
+-------------+------------+-----------+---------------------------+-------------------------------+--------------+
5 rows in set (0.0005 sec)

MySQL  localhost:3306 ssl  ecommerce_db  SQL  select * from Order;
+----------+-------------+---------------------+--------------+
| order_id | customer_id | order_date          | total_amount |
+----------+-------------+---------------------+--------------+
|        1 |           1 | 2025-05-01 10:00:00 |      1226.49 |
|        2 |           2 | 2025-05-01 14:30:00 |       375.99 |
|        3 |           1 | 2025-05-02 09:15:00 |        79.95 |
|        4 |           3 | 2025-05-03 18:00:00 |       129.75 |
|        5 |           4 | 2025-05-04 11:45:00 |       350.00 |
+----------+-------------+---------------------+--------------+
5 rows in set (0.0004 sec)
```

Create tables: Customers and Orders

```
MySQL  localhost:3306 ssl  ecommerce_db  SQL
```

```
CREATE TABLE Customers (
    customer_id INT AUTO_INCREMENT PRIMARY KEY,
    first_name VARCHAR(50) NOT NULL,
    last_name VARCHAR(50) NOT NULL,
    email VARCHAR(100) UNIQUE NOT NULL,
    address VARCHAR(255),
    phone_number VARCHAR(20)
);
```

```
Query OK, 0 rows affected (0.0069 sec)
```

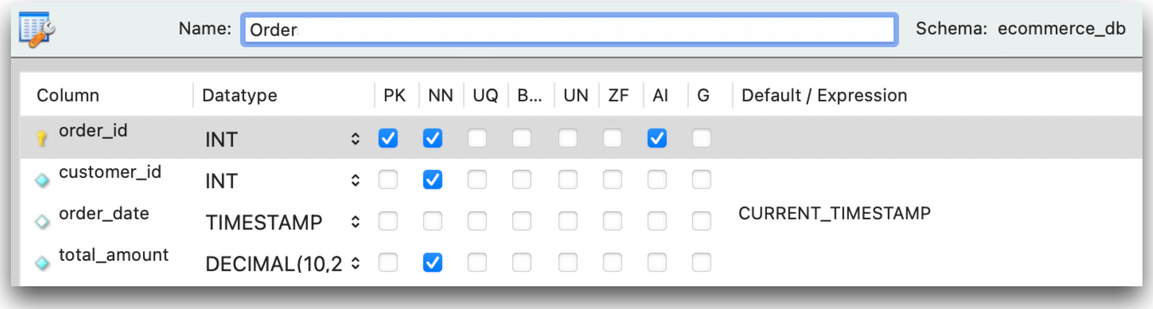
```
MySQL  localhost:3306 ssl  ecommerce_db  SQL
```

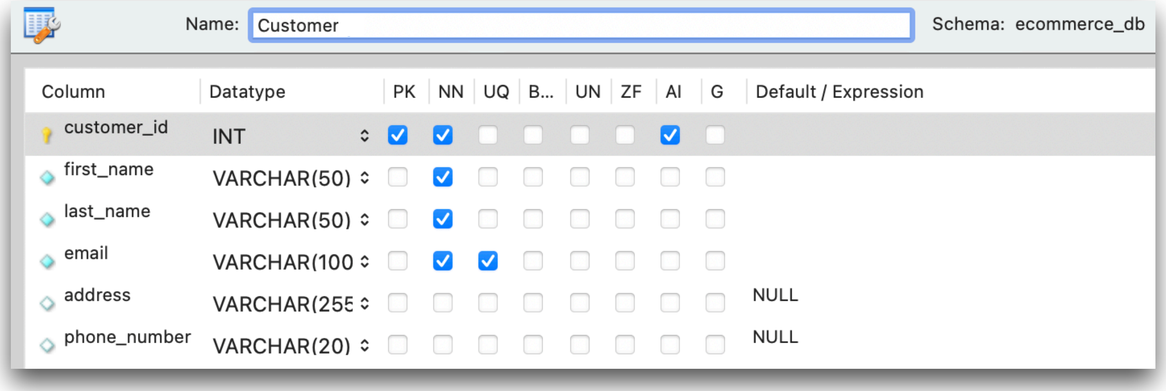
```
CREATE TABLE Orders(
order_id INT PRIMARY KEY AUTO_INCREMENT,
customer_id INT,
order_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
FOREIGN KEY(customer_id) REFERENCES Customers(customer_id)
);
```

Insert Values: Customers and Orders

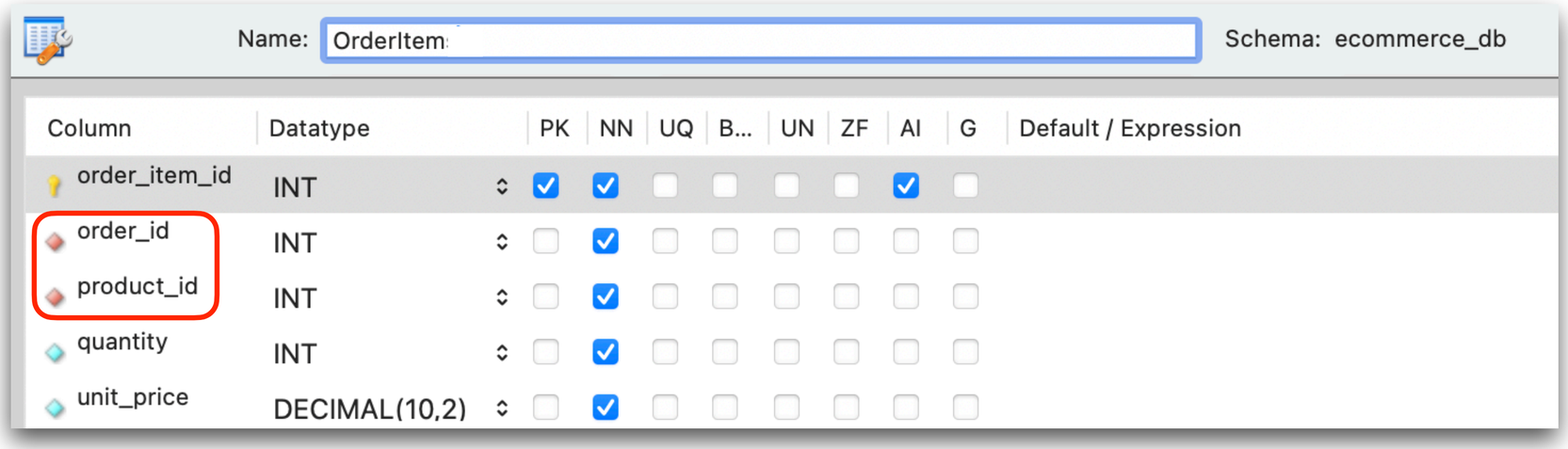
```
MySQL  localhost:3306 ssl  ecommerce_db  SQL  INSERT INTO Customers (first_name, last_name, email, address, phone_number) VALUES
('Alice', 'Smith', 'alice.smith@example.com', '123 Main St, Anytown, CA', '555-1234'),
('Bob', 'Johnson', 'bob.johnson@example.com', '456 Oak Ave, Someville, NY', '555-5678'),
('Charlie', 'Brown', 'charlie.brown@example.com', '789 Pine Ln, Othercity, TX', '555-9012'),
('Diana', 'Garcia', 'diana.garcia@example.com', '101 Elm Rd, Anytown, CA', '555-3456'),
('Eve', 'Williams', 'eve.williams@example.com', '222 Maple Dr, Someville, NY', '555-7890');
Query OK, 5 rows affected (0.0021 sec)

Records: 5  Duplicates: 0  Warnings: 0

 MySQL  localhost:3306 ssl  ecommerce_db  SQL  INSERT INTO Orders (customer_id, order_date, total_amount) VALUES

(1, '2025-05-01 10:00:00', 1226.49), -- Alice ordered something
(2, '2025-05-01 14:30:00', 375.99),  -- Bob ordered something
(1, '2025-05-02 09:15:00', 79.95),   -- Alice ordered something else
(3, '2025-05-03 18:00:00', 129.75),  -- Charlie ordered something

(4, '2025-05-04 11:45:00', 350.00);  -- Diana ordered something
Query OK, 5 rows affected (0.0023 sec)

Records: 5  Duplicates: 0  Warnings: 0
Query OK, 0 rows affected (0.0002 sec)
 MySQL  localhost:3306 ssl  ecommerce_db  SQL
```

```
MySQL  localhost:3306 ssl  ecommerce_db  SQL  select * from Customers;
+-------------+------------+-----------+---------------------------+-------------------------------+--------------+
| customer_id | first_name | last_name | email                     | address                       | phone_number |
+-------------+------------+-----------+---------------------------+-------------------------------+--------------+
|           1 | Alice      | Smith     | alice.smith@example.com   | 123 Main St, Anytown, CA      | 555-1234     |
|           2 | Bob        | Johnson   | bob.johnson@example.com   | 456 Oak Ave, Someville, NY    | 555-5678     |
|           3 | Charlie    | Brown     | charlie.brown@example.com | 789 Pine Ln, Othercity, TX    | 555-9012     |
|           4 | Diana      | Garcia    | diana.garcia@example.com  | 101 Elm Rd, Anytown, CA       | 555-3456     |
|           5 | Eve        | Williams  | eve.williams@example.com  | 222 Maple Dr, Someville, NY   | 555-7890     |
+-------------+------------+-----------+---------------------------+-------------------------------+--------------+
5 rows in set (0.0005 sec)

MySQL  localhost:3306 ssl  ecommerce_db  SQL  select * from Orders;
+----------+-------------+---------------------+--------------+
| order_id | customer_id | order_date          | total_amount |
+----------+-------------+---------------------+--------------+
|        1 |           1 | 2025-05-01 10:00:00 |      1226.49 |
|        2 |           2 | 2025-05-01 14:30:00 |       375.99 |
|        3 |           1 | 2025-05-02 09:15:00 |        79.95 |
|        4 |           3 | 2025-05-03 18:00:00 |       129.75 |
|        5 |           4 | 2025-05-04 11:45:00 |       350.00 |
+----------+-------------+---------------------+--------------+
5 rows in set (0.0004 sec)
```






```
MySQL  localhost:3306 ssl  ecommerce_db  SQL  CREATE TABLE OrderItem (
                                                    order_item_id INT AUTO_INCREMENT PRIMARY KEY,
                                                    order_id INT NOT NULL,
                                                    product_id INT NOT NULL,
                                                    quantity INT NOT NULL,
                                                    unit_price DECIMAL(10, 2) NOT NULL,
                                                    FOREIGN KEY (order_id) REFERENCES OrderDetail(order_id),
                                                    FOREIGN KEY (product_id) REFERENCES Product(product_id)
                                                );
Query OK, 0 rows affected (0.0114 sec)

MySQL  localhost:3306 ssl  ecommerce_db  SQL  DESC Customers;
```



```
MySQL  localhost:3306 ssl  ecommerce_db  SQL  INSERT INTO OrderItem (order_id, product_id, quantity, unit_price) VALUES
                                              (1, 1, 1, 1200.50), -- Order 1 (Alice) bought 1 Laptop
                                              (1, 2, 1, 25.99),   -- Order 1 (Alice) bought 1 Wireless Mouse
                                              (2, 4, 1, 350.00),  -- Order 2 (Bob) bought 1 4K Monitor
                                              (3, 3, 1, 79.95),   -- Order 3 (Alice) bought 1 External Hard Drive
                                              (4, 5, 1, 129.75),  -- Order 4 (Charlie) bought 1 Mechanical Keyboard
                                              (1, 3, 2, 79.95),   -- Order 1 (Alice) bought 2 External Hard Drives
                                              (5, 4, 1, 350.00),  -- Order 5 (Diana) bought 1 4K Monitor
                                              (2, 2, 2, 25.99);   -- Order 2 (Bob) bought 2 Wireless Mice
Query OK, 8 rows affected (0.0039 sec)
```

```
 MySQL  localhost:3306 ssl  ecommerce_db  SQL  select * from OrderItem;
+---------------+----------+------------+----------+------------+
| order_item_id | order_id | product_id | quantity | unit_price |
+---------------+----------+------------+----------+------------+
|             1 |        1 |          1 |        1 |    1200.50 |
|             2 |        1 |          2 |        1 |      25.99 |
|             3 |        2 |          4 |        1 |     350.00 |
|             4 |        3 |          3 |        1 |      79.95 |
|             5 |        4 |          5 |        1 |     129.75 |
|             6 |        1 |          3 |        2 |      79.95 |
|             7 |        5 |          4 |        1 |     350.00 |
|             8 |        2 |          2 |        2 |      25.99 |
+---------------+----------+------------+----------+------------+
8 rows in set (0.0005 sec)
```

Products

product_id	product_name	price
1	Laptop	1200.50
2	Wireless Mouse	25.99
3	External Hard Drive (1TB)	79.95
4	4K Monitor	350.00
5	Mechanical Keyboard	129.75

Customers

customer_id	first_name	last_name	email	address	phone_number
1	Alice	Smith	alice.smith@example.com	123 Main St, Anytown, CA	555-1234
2	Bob	Johnson	bob.johnson@example.com	456 Oak Ave, Someville, NY	555-5678
3	Charlie	Brown	charlie.brown@example.com	789 Pine Ln, Othercity, TX	555-9012
4	Diana	Garcia	diana.garcia@example.com	101 Elm Rd, Anytown, CA	555-3456
5	Eve	Williams	eve.williams@example.com	222 Maple Dr, Someville, NY	555-7890

OrderDetail

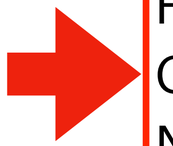
order_id	customer_id	order_date	total_amount
1	1	2025-05-01 10:00:00	1226.49
2	2	2025-05-01 14:30:00	375.99
3	1	2025-05-02 09:15:00	79.95
4	3	2025-05-03 18:00:00	129.75
5	4	2025-05-04 11:45:00	350.00

OrderItem

order_item_id	order_id	product_id	quantity	unit_price
1	1	1	1	1200.50
2	1	2	1	25.99
3	2	4	1	350.00
4	3	3	1	79.95
5	4	5	1	129.75
6	1	3	2	79.95
7	5	4	1	350.00
8	2	2	2	25.99

What is a Composite Key?

Composite Keys: Defining Uniqueness



Candidate Key
Primary Key
Foreign Key
Composite Key
Natural Key
Surrogate Key

- A composite key combines two or more columns to uniquely identify each row in a table.
- Used when a single column cannot guarantee uniqueness.
- The combination of columns must be unique for each record.
- Enhances data integrity and retrieval efficiency.

```
MySQL  localhost:3306 ssl  bank_db  SQL  SELECT * FROM account;
+---------+----------+----------------------+
| acc_num | acc_type | acc_descr            |
+---------+----------+----------------------+
|     101 |        1 | Checking Account     |
|     101 |        2 | Savings Account      |
|     102 |        1 | Checking Account     |
|     201 |        1 | Checking Account     |
|     201 |        3 | Money Market Account |
+---------+----------+----------------------+
```

Creating Composite Keys in SQL

SQL Syntax for Composite Keys

To create a composite key in SQL, use the PRIMARY KEY constraint followed by the columns that will form the composite key.

```
+---------+----------+----------------------+
| acc_num | acc_type | acc_descr            |
+---------+----------+----------------------+
|     101 |        1 | Checking Account     |
|     101 |        2 | Savings Account      |
|     102 |        1 | Checking Account     |
|     201 |        1 | Checking Account     |
|     201 |        3 | Money Market Account |
+---------+----------+----------------------+
5 rows in set (0.0004 sec)
```

```
MySQL  localhost:3306 ssl  bank_db  SQL  CREATE TABLE Account (
                                             acc_num INTEGER,
                                             acc_type INTEGER,
                                             acc_descr VARCHAR(20),
                                             PRIMARY KEY (acc_num, acc_type)
                                         );
Query OK, 0 rows affected (0.0080 sec)
```

Creating Composite Keys in SQL

SQL Syntax for Composite Keys

```
+---------+----------+----------------------+
| acc_num | acc_type | acc_descr            |
+---------+----------+----------------------+
|     101 |        1 | Checking Account     |
|     101 |        2 | Savings Account      |
|     102 |        1 | Checking Account     |
|     201 |        1 | Checking Account     |
|     201 |        3 | Money Market Account |
+---------+----------+----------------------+
5 rows in set (0.0004 sec)
```

```
CREATE TABLE Account (
    acc_num INTEGER,
    acc_type INTEGER,
    acc_descr VARCHAR(20),
    PRIMARY KEY (acc_num, acc_type)
);
```

```
MySQL  localhost:3306 ssl  bank_db  SQL
```

```
INSERT INTO Account (acc_num, acc_type, acc_descr) VALUES
(101, 1, 'Checking Account'),
(101, 2, 'Savings Account'),
(102, 1, 'Checking Account'),
(201, 1, 'Checking Account'),
(201, 3, 'Money Market Account');
Query OK, 5 rows affected (0.0024 sec)
```

Show Structure of Table

DESC TABLE_NAME;

```
MySQL  localhost:3306 ssl  bank_db  SQL > DESC Account;
+-----------+-------------+------+-----+---------+-------+
| Field     | Type        | Null | Key | Default | Extra |
+-----------+-------------+------+-----+---------+-------+
| acc_num   | int         | NO   | PRI | NULL    |       |
| acc_type  | int         | NO   | PRI | NULL    |       |
| acc_descr | varchar(20) | YES  |     | NULL    |       |
+-----------+-------------+------+-----+---------+-------+
```

Natural Keys

The Inherent Identifiers

- Natural keys are attributes or combinations of attributes that already exist in the data and can uniquely pinpoint a record.
- They are called “natural” simply because they arise from the inherent properties of the entity we are modeling.

Candidate Key
Primary Key
Foreign Key
Composite Key
Natural Key
Surrogate Key

```
MySQL  localhost:3306 ssl  admission_db  SQL > SELECT * FROM Student;
+-------------+--------------+------------+-----------------+-----------------+------+
| national_id | name         | birth_date | enrollment_date | graduation_date | gpa  |
+-------------+--------------+------------+-----------------+-----------------+------+
| 12345678901 | Alice Smith  | 2002-05-15 | 2020-09-01      | 2024-05-10      | 3.85 |
| 23456789012 | Fiona Green  | 2003-07-18 | 2021-09-01      | NULL            | 3.78 |
| 34567890123 | Hannah Black | 2001-12-11 | 2019-09-01      | 2023-05-12      |  3.8 |
| 43210987654 | Diana Miller | 2002-11-05 | 2020-09-01      | 2024-05-10      | 3.92 |
| 56789012345 | Charlie Brown| 2003-01-10 | 2021-09-01      | NULL            |  3.2 |
| 65432109876 | Isaac Grey   | 2003-04-08 | 2021-09-01      | NULL            | 3.45 |
| 76543210987 | George White | 2002-09-03 | 2020-09-01      | 2024-05-10      |  3.5 |
| 87654321098 | Ethan Davis  | 2001-03-28 | 2019-09-01      | 2023-05-12      | 3.65 |
| 90123456789 | Julia Purple | 2002-06-20 | 2020-09-01      | 2024-05-10      |    4 |
| 98765432109 | Bob Johnson  | 2001-08-22 | 2019-09-01      | 2023-05-12      |  3.4 |
+-------------+--------------+------------+-----------------+-----------------+------+
```

```
MySQL localhost:3306 ssl admission_db SQL
```

```
CREATE TABLE Student (
    national_id BIGINT PRIMARY KEY,
    name VARCHAR(60),
    birth_date DATE,
    enrollment_date DATE,
    graduation_date DATE,
    gpa FLOAT
);
```

```
Query OK, 0 rows affected (0.0067 sec)
```

```
MySQL localhost:3306 ssl admission_db SQL
```

```
INSERT INTO Student (national_id, name, birth_date, enrollment_date,
graduation_date, gpa) VALUES
(12345678901, 'Alice Smith', '2002-05-15', '2020-09-01', '2024-05-10', 3.85),
(98765432109, 'Bob Johnson', '2001-08-22', '2019-09-01', '2023-05-12', 3.40),
(56789012345, 'Charlie Brown', '2003-01-10', '2021-09-01', NULL, 3.20),
(43210987654, 'Diana Miller', '2002-11-05', '2020-09-01', '2024-05-10', 3.92),
(87654321098, 'Ethan Davis', '2001-03-28', '2019-09-01', '2023-05-12', 3.65),
(23456789012, 'Fiona Green', '2003-07-18', '2021-09-01', NULL, 3.78),
(76543210987, 'George White', '2002-09-03', '2020-09-01', '2024-05-10', 3.50),
(34567890123, 'Hannah Black', '2001-12-11', '2019-09-01', '2023-05-12', 3.80),
(65432109876, 'Isaac Grey', '2003-04-08', '2021-09-01', NULL, 3.45),
(90123456789, 'Julia Purple', '2002-06-20', '2020-09-01', '2024-05-10', 4.00);
```

```
Query OK, 10 rows affected (0.0030 sec)

Records: 10  Duplicates: 0  Warnings: 0
```

```
MySQL localhost:3306 ssl admission_db SQL
```

```
MySQL  localhost:3306 ssl  admission_db  SQL > SELECT * FROM Student;
+-------------+---------------+------------+-----------------+-----------------+------+
| national_id | name          | birth_date | enrollment_date | graduation_date | gpa  |
+-------------+---------------+------------+-----------------+-----------------+------+
| 12345678901 | Alice Smith   | 2002-05-15 | 2020-09-01      | 2024-05-10      | 3.85 |
| 23456789012 | Fiona Green   | 2003-07-18 | 2021-09-01      | NULL            | 3.78 |
| 34567890123 | Hannah Black  | 2001-12-11 | 2019-09-01      | 2023-05-12      |  3.8 |
| 43210987654 | Diana Miller  | 2002-11-05 | 2020-09-01      | 2024-05-10      | 3.92 |
| 56789012345 | Charlie Brown | 2003-01-10 | 2021-09-01      | NULL            |  3.2 |
| 65432109876 | Isaac Grey    | 2003-04-08 | 2021-09-01      | NULL            | 3.45 |
| 76543210987 | George White  | 2002-09-03 | 2020-09-01      | 2024-05-10      |  3.5 |
| 87654321098 | Ethan Davis   | 2001-03-28 | 2019-09-01      | 2023-05-12      | 3.65 |
| 90123456789 | Julia Purple  | 2002-06-20 | 2020-09-01      | 2024-05-10      |    4 |
| 98765432109 | Bob Johnson   | 2001-08-22 | 2019-09-01      | 2023-05-12      |  3.4 |
+-------------+---------------+------------+-----------------+-----------------+------+
```

Surrogate Keys

The Artificial Alternative

Candidate Key
Primary Key
Foreign Key
Composite Key
Natural Key
Surrogate Key

- Surrogate keys, conversely, are artificial identifiers, typically auto-generated numbers or globally unique identifiers, created solely to uniquely identify records.
- They have no inherent meaning in the real sense. They are “surrogates” in the sense that they stand in for natural keys, often providing a simpler yet more stable way to identify records.

```
MySQL  localhost:3306 ssl  admission_db  SQL  select * from Student;
+----+-------------+---------------+------------+-----------------+-----------------+------+
| id | national_id | name          | birth_date | enrollment_date | graduation_date | gpa  |
+----+-------------+---------------+------------+-----------------+-----------------+------+
|  1 | 12345678901 | Alice Smith   | 2002-05-15 | 2020-09-01      | 2024-05-10      | 3.85 |
|  2 | 98765432109 | Bob Johnson   | 2001-08-22 | 2019-09-01      | 2023-05-12      |  3.4 |
|  3 | 56789012345 | Charlie Brown | 2003-01-10 | 2021-09-01      | NULL            |  3.2 |
|  4 | 43210987654 | Diana Miller  | 2002-11-05 | 2020-09-01      | 2024-05-10      | 3.92 |
|  5 | 87654321098 | Ethan Davis   | 2001-03-28 | 2019-09-01      | 2023-05-12      | 3.65 |
|  6 | 23456789012 | Fiona Green   | 2003-07-18 | 2021-09-01      | NULL            | 3.78 |
|  7 | 76543210987 | George White  | 2002-09-03 | 2020-09-01      | 2024-05-10      |  3.5 |
|  8 | 34567890123 | Hannah Black  | 2001-12-11 | 2019-09-01      | 2023-05-12      |  3.8 |
|  9 | 65432109876 | Isaac Grey    | 2003-04-08 | 2021-09-01      | NULL            | 3.45 |
| 10 | 90123456789 | Julia Purple  | 2002-06-20 | 2020-09-01      | 2024-05-10      |    4 |
```

```
MySQL localhost:3306 ssl admission_db SQL > CREATE TABLE Student (
                                                id INT AUTO_INCREMENT PRIMARY KEY,
                                                national_id BIGINT UNIQUE,
                                                name VARCHAR(60),
                                                birth_date DATE,
                                                enrollment_date DATE,
                                                graduation_date DATE,
                                                gpa FLOAT
                                            );
Query OK, 0 rows affected (0.0083 sec)
MySQL localhost:3306 ssl admission_db SQL > INSERT INTO Student (national_id, name, birth_date, enrollment_date, graduation_date,
gpa) VALUES
                                            (12345678901, 'Alice Smith', '2002-05-15', '2020-09-01', '2024-05-10', 3.85),
                                            (98765432109, 'Bob Johnson', '2001-08-22', '2019-09-01', '2023-05-12', 3.40),
                                            (56789012345, 'Charlie Brown', '2003-01-10', '2021-09-01', NULL, 3.20),
                                            (43210987654, 'Diana Miller', '2002-11-05', '2020-09-01', '2024-05-10', 3.92),
                                            (87654321098, 'Ethan Davis', '2001-03-28', '2019-09-01', '2023-05-12', 3.65),
                                            (23456789012, 'Fiona Green', '2003-07-18', '2021-09-01', NULL, 3.78),
                                            (76543210987, 'George White', '2002-09-03', '2020-09-01', '2024-05-10', 3.50),
                                            (34567890123, 'Hannah Black', '2001-12-11', '2019-09-01', '2023-05-12', 3.80),
                                            (65432109876, 'Isaac Grey', '2003-04-08', '2021-09-01', NULL, 3.45),
                                            (90123456789, 'Julia Purple', '2002-06-20', '2020-09-01', '2024-05-10', 4.00);
Query OK, 10 rows affected (0.0039 sec)

Records: 10  Duplicates: 0  Warnings: 0
MySQL localhost:3306 ssl admission_db SQL > select * from Student;
+----+-------------+---------------+------------+-----------------+-----------------+------+
| id | national_id | name          | birth_date | enrollment_date | graduation_date | gpa  |
+----+-------------+---------------+------------+-----------------+-----------------+------+
|  1 | 12345678901 | Alice Smith   | 2002-05-15 | 2020-09-01      | 2024-05-10      | 3.85 |
|  2 | 98765432109 | Bob Johnson   | 2001-08-22 | 2019-09-01      | 2023-05-12      |  3.4 |
|  3 | 56789012345 | Charlie Brown | 2003-01-10 | 2021-09-01      | NULL            |  3.2 |
|  4 | 43210987654 | Diana Miller  | 2002-11-05 | 2020-09-01      | 2024-05-10      | 3.92 |
|  5 | 87654321098 | Ethan Davis   | 2001-03-28 | 2019-09-01      | 2023-05-12      | 3.65 |
|  6 | 23456789012 | Fiona Green   | 2003-07-18 | 2021-09-01      | NULL            | 3.78 |
|  7 | 76543210987 | George White  | 2002-09-03 | 2020-09-01      | 2024-05-10      |  3.5 |
|  8 | 34567890123 | Hannah Black  | 2001-12-11 | 2019-09-01      | 2023-05-12      |  3.8 |
|  9 | 65432109876 | Isaac Grey    | 2003-04-08 | 2021-09-01      | NULL            | 3.45 |
| 10 | 90123456789 | Julia Purple  | 2002-06-20 | 2020-09-01      | 2024-05-10      |    4 |
+----+-------------+---------------+------------+-----------------+-----------------+------+
```

Integrity Constraints

Types with Examples

Integrity constraints in DBMS are sets of rules that keep your database accurate, consistent, and secure.

They ensure each data entry, update, or deletion follows predefined standards, so you always work with information you can trust.

1. Entity Integrity
2. Referential Integrity
3. Domain Constraints
4. General Constraints

Integrity Constraints

Types with Examples

- Entity Integrity
  - Ensures each row in a table has a unique and non-null primary key.
  - Prevents duplicate rows and ensures each record is identifiable.

```
MySQL  localhost:3306 ssl  admission_db  SQL
```

```
CREATE TABLE Student (
 student_id INT PRIMARY KEY,
 name VARCHAR(50)
);
```

student_id must have a value (not NULL) for every student.

Integrity Constraints

Types with Examples

- Referential Integrity
  - Maintains consistency between related tables through foreign keys.
  - A foreign key must match an existing primary key value in another table (or be NULL, if allowed).

Integrity Constraints

Types with Examples

- Referential Integrity

```
MySQL  localhost:3306 ssl  miu_db  SQL  CREATE TABLE Department (
                                         dept_id INT PRIMARY KEY,
                                         dept_name VARCHAR(50)
                                     );
Query OK, 0 rows affected (0.0055 sec)
MySQL  localhost:3306 ssl  miu_db  SQL  CREATE TABLE Employee (
                                         emp_id INT PRIMARY KEY,
                                         emp_name VARCHAR(50),
                                         dept_id INT,
                                         FOREIGN KEY (dept_id) REFERENCES Department(dept_id)
                                     );
```

dept_id in Employee must match a dept_id in Department.

Integrity Constraints

Types with Examples

- Domain Constraint
  - Ensure that the value of an attribute (column) falls within a specified domain (set of valid values).

```
CREATE TABLE Product (
  product_id INT PRIMARY KEY,
  price DECIMAL(10, 2) CHECK (price > 0),
  category VARCHAR(20) CHECK (category IN ('Electronics', 'Clothing', 'Food')),
  type VARCHAR(20) CHECK (type IN ('Consumable', 'Non-Consumable'))
);
```

Integrity Constraints

Types with Examples

- General Constraints
  - Additional rules specified by users or database administrators that define or constrain some aspect of the enterprise.
    - E.g. A branch may have only 20 staff members

```
MySQL  localhost:3306 ssl  bank_db  SQL > CREATE TABLE Branch (
                                         branch_id INT PRIMARY KEY,
                                         branch_name VARCHAR(100)
                                     );
Query OK, 0 rows affected (0.0047 sec)
MySQL  localhost:3306 ssl  bank_db  SQL > CREATE TABLE Staff (
                                         staff_id INT PRIMARY KEY,
                                         name VARCHAR(100),
                                         branch_id INT,
                                         FOREIGN KEY (branch_id) REFERENCES Branch(branch_id)
                                     );
Query OK, 0 rows affected (0.0088 sec)
MySQL  localhost:3306 ssl  bank_db  SQL > DELIMITER //
MySQL  localhost:3306 ssl  bank_db  SQL > CREATE TRIGGER limit_staff_per_branch
                                     BEFORE INSERT ON Staff
                                     FOR EACH ROW
                                     BEGIN
                                         DECLARE staff_count INT;

                                         SELECT COUNT(*) INTO staff_count
                                         FROM Staff
                                         WHERE branch_id = NEW.branch_id;

                                         IF staff_count >= 20 THEN
                                             SIGNAL SQLSTATE '45000'
                                             SET MESSAGE_TEXT = 'A branch cannot have more than 20 staff members';
                                         END IF;
                                     END;
                                     //DELIMITER ;
Query OK, 0 rows affected (0.0027 sec)
MySQL  localhost:3306 ssl  bank_db  SQL >
```

SIGNAL is a MySQL statement used to raise (throw) an error intentionally inside triggers, stored procedures, or functions.

SQLSTATE is a standard code that describes the type of SQL error.

It allows you to abort an operation and provide a custom error message.

Main Point

A database can be modeled as a set of relations. It is always advantageous to find a simple basis for a complex field because it provides a way to manage the complexity of the field.

Science & Technology of Consciousness: Vedic Science has discovered that the simplest form of awareness is the basis for all of the creation.

Functional Dependencies

- A functional dependency is a relationship between two sets of attributes in a database, where one set (the determinant) determines the values of the other set (the dependent).
- For example, in a database of employees, the employee ID number (determinant) would determine the employee's name, address, and other personal information (dependent). This means that, given an employee ID number, we can determine the corresponding employee's name and other personal information, but not vice versa.

EmployeeID	EmployeeName	Address	Department
E001	Alice Smith	123 Main St	HR
E002	Bob Johnson	456 Oak Ave	IT
E003	Charlie Brown	789 Pine Ln	IT

Functional Dependencies

EmployeeID	EmployeeName	Address	Department
E001	Alice Smith	123 Main St	HR
E002	Bob Johnson	456 Oak Ave	IT
E003	Charlie Brown	789 Pine Ln	IT

Functional dependencies can also be represented using mathematical notation.

$X \rightarrow Y$

- X is a set of attributes (determinant)
- Y is a set of attributes (dependent)

EmployeeID → EmployeeName
EmployeeID → Address
EmployeeID → Department
EmployeeID → EmployeeName, Address, Department

Characteristics of Functional Dependencies

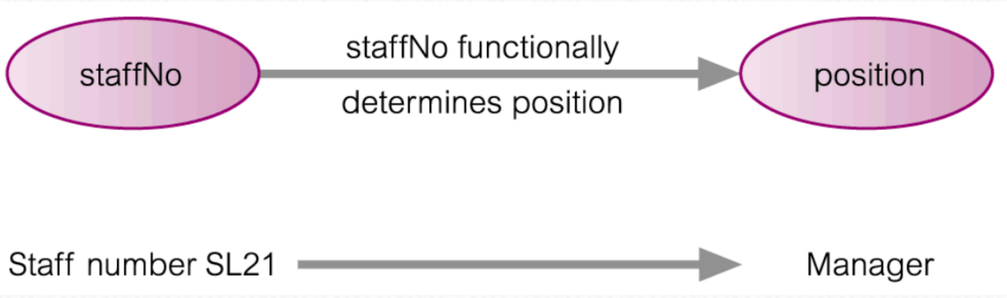
$X \rightarrow Y$

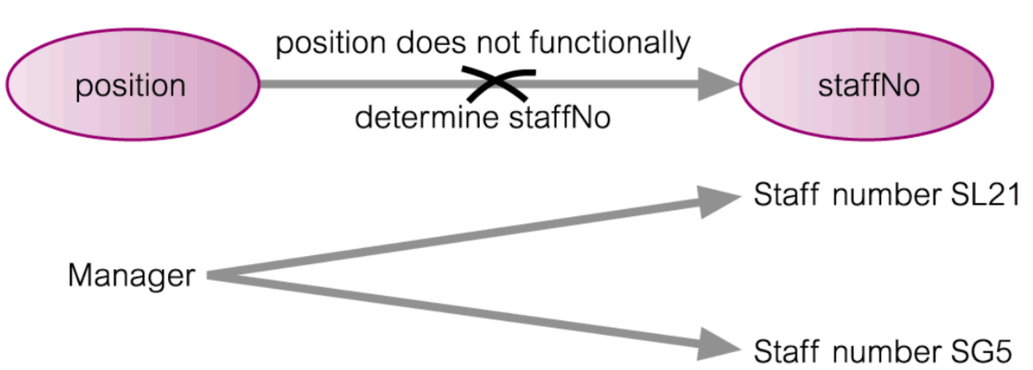
- There is a one-to-one relationship between the attribute(s) on the left-hand side (determinant) and those on the right-hand side (dependent) of a functional dependency.
- Holds for all time.
- The determinant has the minimal number of attributes necessary to maintain the dependency with the attribute(s) on the right hand-side.
  - This requirement is called full functional dependency.

An Example of Functional Dependency

Staff Branch

staffNo	sName	position	salary	branchNo	bAddress
SL21	John White	Manager	30000	B005	22 Deer Rd, London
SG37	Ann Beech	Assistant	12000	B003	163 Main St, Glasgow
SG14	David Ford	Supervisor	18000	B003	163 Main St, Glasgow
SA9	Mary Howe	Assistant	9000	B007	16 Argyll St, Aberdeen
SG5	Susan Brand	Manager	24000	B003	163 Main St, Glasgow
SL41	Julie Lee	Assistant	9000	B005	22 Deer Rd, London





Note:
There is a one-to-one relationship between the attribute(s) on the left-hand side (determinant) and those on the right-hand side (dependent) of a functional dependency.

Example Functional Dependency that holds true for all Time

- Based on sample data, the following functional dependencies appear to hold.
  - staffNo → sName
  - sName → staffNo
- However, the only functional dependency that remains true for all possible values for the staffNo and sName attributes of the Staff relation is:
  - staffNo → sName

Staff Branch

staffNo	sName	position	salary	branchNo	bAddress
SL21	John White	Manager	30000	B005	22 Deer Rd, London
SG37	Ann Beech	Assistant	12000	B003	163 Main St, Glasgow
SG14	David Ford	Supervisor	18000	B003	163 Main St, Glasgow
SA9	Mary Howe	Assistant	9000	B007	16 Argyll St, Aberdeen
SG5	Susan Brand	Manager	24000	B003	163 Main St, Glasgow
SL41	Julie Lee	Assistant	9000	B005	22 Deer Rd, London

Example - Identify FDs for the StaffBranch Relation

- Assume that there is only one branch at a particular address.

staffNo	sName	position	salary	branchNo	bAddress
SL21	John White	Manager	30000	B005	22 Deer Rd, London
SG37	Ann Beech	Assistant	12000	B003	163 Main St, Glasgow
SG14	David Ford	Supervisor	18000	B003	163 Main St, Glasgow
SA9	Mary Howe	Assistant	9000	B007	16 Argyll St, Aberdeen
SG5	Susan Brand	Manager	24000	B003	163 Main St, Glasgow
SL41	Julie Lee	Assistant	9000	B005	22 Deer Rd, London

Example - Identify FDs for the StaffBranch Relation

- staffNo → sName, position, salary, branchNo, bAddress
- branchNo → bAddress
- bAddress → branchNo
- branchNo, position → salary
- bAddress, position → salary

staffNo	sName	position	salary	branchNo	bAddress
SL21	John White	Manager	30000	B005	22 Deer Rd, London
SG37	Ann Beech	Assistant	12000	B003	163 Main St, Glasgow
SG14	David Ford	Supervisor	18000	B003	163 Main St, Glasgow
SA9	Mary Howe	Assistant	9000	B007	16 Argyll St, Aberdeen
SG5	Susan Brand	Manager	24000	B003	163 Main St, Glasgow
SL41	Julie Lee	Assistant	9000	B005	22 Deer Rd, London

Full Functional Dependency

- Indicates that if A and B are attributes of a relation, B is fully functionally dependent on A if B is functionally dependent on A, but not on any proper subset of A.

Staff Branch

staffNo	sName	position	salary	branchNo	bAddress
SL21	John White	Manager	30000	B005	22 Deer Rd, London
SG37	Ann Beech	Assistant	12000	B003	163 Main St, Glasgow
SG14	David Ford	Supervisor	18000	B003	163 Main St, Glasgow
SA9	Mary Howe	Assistant	9000	B007	16 Argyll St, Aberdeen
SG5	Susan Brand	Manager	24000	B003	163 Main St, Glasgow
SL41	Julie Lee	Assistant	9000	B005	22 Deer Rd, London

Full Functional Dependency

- Full functionally dependency or partial functional dependency?
  - staffNo, sName -> branchNo

Staff Branch

staffNo	sName	position	salary	branchNo	bAddress
SL21	John White	Manager	30000	B005	22 Deer Rd, London
SG37	Ann Beech	Assistant	12000	B003	163 Main St, Glasgow
SG14	David Ford	Supervisor	18000	B003	163 Main St, Glasgow
SA9	Mary Howe	Assistant	9000	B007	16 Argyll St, Aberdeen
SG5	Susan Brand	Manager	24000	B003	163 Main St, Glasgow
SL41	Julie Lee	Assistant	9000	B005	22 Deer Rd, London

Answer:

Partial functional dependency

branchNo is also functionally dependent on a subset of (satffNo, sName), namely staffNo

Example of Full Functional Dependency

STOCKS (Symbol, Company, Headquarters, Date, ClosePrice)

Symbol	Company	Headquarters	Date	ClosePrice
MSFT	Microsoft	Redmond, WA	09/07/2023	23.96
MSFT	Microsoft	Redmond, WA	09/08/2023	23.93
MSFT	Microsoft	Redmond, WA	09/09/2023	24.01
ORCL	Oracle	Redwood Shores, CA	09/07/2023	24.27
ORCL	Oracle	Redwood Shores, CA	09/08/2023	24.14
ORCL	Oracle	Redwood Shores, CA	09/09/2023	24.33

- **(Symbol, Date) → ClosePrice** — **FFD**
- **(Symbol, Date) → Company** **X FFD**

Identifying Primary Key for a Relation using FDs

- Main purpose of identifying a set of functional dependencies for a relation is to specify the set of integrity constraints that must hold on a relation.
- An important integrity constraint to consider first is the identification of candidate keys, one of which is selected to be the primary key for the relation.

Identifying Primary Key for a Relation using FDs

- staffNo → sName, position, salary, branchNo, bAddress
- branchNo → bAddress
- bAddress → branchNo
- branchNo, position → salary
- bAddress, position → salary

staffNo	sName	position	salary	branchNo	bAddress
SL21	John White	Manager	30000	B005	22 Deer Rd, London
SG37	Ann Beech	Assistant	12000	B003	163 Main St, Glasgow
SG14	David Ford	Supervisor	18000	B003	163 Main St, Glasgow
SA9	Mary Howe	Assistant	9000	B007	16 Argyll St, Aberdeen
SG5	Susan Brand	Manager	24000	B003	163 Main St, Glasgow
SL41	Julie Lee	Assistant	9000	B005	22 Deer Rd, London

The only candidate key and therefore primary key for StaffBranch relation, is staffNo, as all other attributes of the relation are functionally dependent on staffNo.

The Process of Normalization

- Formal technique for analyzing a relation based on its primary key and the functional dependencies between the attributes of that relation.

- Often executed as a series of steps. Each step corresponds to a specific normal form, which has known properties.

Example of Unnormalized Form (UNF)

- Let's consider a scenario of a Student Registration System.

Student_ID	Student_Name	Course_IDs	Course_Names	Instructor	Instructor_Contact
S101	John Doe	C001, C002	Math, Physics	Dr. Smith, Dr. Brown	12345, 67890
S102	Alice Smith	C001	Math	Dr. Smith	12345

Denormalized Table

Ref: https://mum0-my.sharepoint.com/:x:/g/personal/bvarghese_miu_edu/ERGk4EnGw_5Ai3w4EqaxjvUBwojI9kah970ugydRvzRtCA?e=Qrd3Ai

Student_ID	Student_Name	Course_IDs	Course_Names	Instructor	Instructor_Contact
S101	John Doe	C001, C002	Math, Physics	Dr. Smith, Dr. Brown	12345, 67890
S102	Alice Smith	C001	Math	Dr. Smith	12345

❌ Repeating Groups – Multiple values in Course_IDs, Course_Names, Instructor, and Instructor_Contact fields.
❌ Data Redundancy – Instructor details are repeated for each student.
❌ Difficult to Query – Searching for students in Physics requires parsing multiple values in Course_Names.

1 NF

- Each column contains atomic values (no multiple values in a single field).
- Each column has unique values (no repeating groups or arrays).
- Each row is uniquely identifiable (using a primary key).

Repeating Groups

OrderID	CustomerName	Item1	Quantity1	Item2	Quantity2	Item3	Quantity3
101	Alice	Book	1	Pen	2		
102	Bob	Shirt	1				
103	Charlie	Mug	2	Hat	1	Socks	3

1 NF

Each column contains atomic values (no multiple values in a single field).

Each column has unique values (no repeating groups or arrays).

Each row is uniquely identifiable (using a primary key).

Student_ID	Student_Name	Course_IDs	Course_Names	Instructor	Instructor_Cont
S101	John Doe	C001, C002	Math, Physics	Dr. Smith, Dr.	12345, 67890
S102	Alice Smith	C001	Math	Dr. Smith	12345

Student_ID	Student_Name	Course_ID	Course_Name	Instructor	Instructor_Contact
S101	John Doe	C001	Math	Dr. Smith	12345
S101	John Doe	C002	Physics	Dr. Brown	67890
S102	Alice Smith	C001	Math	Dr. Smith	12345

1 NF

- Each column contains atomic values (no multiple values in a single field).
- Each column has unique values (no repeating groups or arrays).
- Each row is uniquely identifiable (using a primary key).

ORDER_NUMBER	QUANTITY_ORDERED	PRICE_EACH	SALES	ORDER_DATE	STATUS	PRODUCT_CODE	PRODUCT_NAME	MSRP	CUSTOMER_NAME	PHONE	ADDRESS_LINE1	ADDRESS_LINE2	CITY	STATE	POSTAL_CODE	COUNTRY	TERRITORY	CONTACT_LAST_NAME	CONTACT_FIRST_NAME	DEAL_SIZE
1001	30	95.7	2871.00	2/24/03	Shipped	S10_1678	Motorcycles	95	Land of Toys Inc.	2125557818	897 Long Airport Avenue		NYC	NY	10022	USA	NA	Yu	Kwai	Small
1002	34	81.35	2765.90	5/7/03	Shipped	S10_1678	Motorcycles	95	Reims Collectables	26.47.1555	59 rue de l'Abbaye		Reims		51100	France	EMEA	Henriot	Paul	Small
1003	41	94.74	3884.34	7/1/03	Shipped	S10_1678	Motorcycles	95	Lyon Souveniers	+33 1 46 62 7555	27 rue du Colonel Pierre Avia		Paris		75508	France	EMEA	Da Cunha	Daniel	Medium
1004	45	83.26	3746.70	8/25/03	Shipped	S10_1678	Motorcycles	95	Toys4GrownUps.com	6265557265	78934 Hillside Dr.		Pasadena	CA	90003	USA	NA	Young	Julie	Medium
1005	49	100	5205.27	10/10/03	Shipped	S10_1678	Motorcycles	95	Corporate Gift Ideas Co.	6505551386	7734 Strong St.		San Francisco	CA		USA	NA	Brown	Julie	Medium
1006	4, 3	800.5, 900.00	5902.00	10/28/03	Shipped	S10_1678, S10_1679	Car, Truck	800, 900	Technics Stores Inc.	6505556809	9408 Furth Circle		Burlingame	CA	94217	USA	NA	Hirano	Juri	Small

ORDER_NUMBER	QUANTITY_ORDERED	PRICE_EACH	SALES	ORDER_DATE	STATUS	PRODUCT_CODE	PRODUCT_NAME	MSRP	CUSTOMER_NAME	PHONE	ADDRESS_LINE1	ADDRESS_LINE2	CITY	STATE	POSTAL_CODE	COUNTRY	TERRITORY	CONTACT_LAST_NAME	CONTACT_FIRST_NAME	DEAL_SIZE
1001	30	95.7	2871.00	2/24/03	Shipped	S10_1678	Motorcycles	95	Land of Toys Inc.	2125557818	897 Long Airport Avenue		NYC	NY	10022	USA	NA	Yu	Kwai	Small
1002	34	81.35	2765.90	5/7/03	Shipped	S10_1678	Motorcycles	95	Reims Collectables	26.47.1555	59 rue de l'Abbaye		Reims		51100	France	EMEA	Henriot	Paul	Small
1003	41	94.74	3884.34	7/1/03	Shipped	S10_1678	Motorcycles	95	Lyon Souveniers	+33 1 46 62 7555	27 rue du Colonel Pierre Avia		Paris		75508	France	EMEA	Da Cunha	Daniel	Medium
1004	45	83.26	3746.70	8/25/03	Shipped	S10_1678	Motorcycles	95	Toys4GrownUps.com	6265557265	78934 Hillside Dr.		Pasadena	CA	90003	USA	NA	Young	Julie	Medium
1005	49	100	5205.27	10/10/03	Shipped	S10_1678	Motorcycles	95	Corporate Gift Ideas Co.	6505551386	7734 Strong St.		San Francisc	CA		USA	NA	Brown	Julie	Medium
1006	4	800.5	3202.00	10/28/03	Shipped	S10_1678	Car	800	Technics Stores Inc.	6505556809	9408 Furth Circle		Burlingame	CA	94217	USA	NA	Hirano	Juri	Small
1006	3	900	2700.00	10/28/03	Shipped	S10_1679	Truck	900	Technics Stores Inc.	6505556809	9408 Furth Circle		Burlingame	CA	94217	USA	NA	Hirano	Juri	Small

ORDER_NUMBER	QUANTITY_ORDERED	PRICE_EACH	SALES	ORDER_DATE	STATUS	PRODUCT_CODE	PRODUCT_NAME	MSRP	CUSTOMER_NAME	PHONE	ADDRESS_LINE1	ADDRESS_LINE2	CITY	STATE	POSTAL_CODE	COUNTRY	TERRITORY	CONTACT_LAST_NAME	CONTACT_FIRST_NAME	DEAL_SIZE
1001	30	95.7	2871.00	2/24/03	Shipped	S10_1678	Motorcycles	95	Land of Toys Inc.	2125557818	897 Long Airport Avenue		NYC	NY	10022	USA	NA	Yu	Kwai	Small
1002	34	81.35	2765.90	5/7/03	Shipped	S10_1678	Motorcycles	95	Reims Collectables	26.47.1555	59 rue de l'Abbaye		Reims		51100	France	EMEA	Henriot	Paul	Small
1003	41	94.74	3884.34	7/1/03	Shipped	S10_1678	Motorcycles	95	Lyon Souveniers	+33 1 46 62 7555	27 rue du Colonel Pierre Avia		Paris		75508	France	EMEA	Da Cunha	Daniel	Medium
1004	45	83.26	3746.70	8/25/03	Shipped	S10_1678	Motorcycles	95	Toys4GrownUps.com	6265557265	78934 Hillside Dr.		Pasadena	CA	90003	USA	NA	Young	Julie	Medium
1005	49	100	5205.27	10/10/03	Shipped	S10_1678	Motorcycles	95	Corporate Gift Ideas Co.	6505551386	7734 Strong St.		San Francisc	CA		USA	NA	Brown	Julie	Medium
1006	4	800.5	3202.00	10/28/03	Shipped	S10_1678	Car	800	Technics Stores Inc.	6505556809	9408 Furth Circle		Burlingame	CA	94217	USA	NA	Hirano	Juri	Small
1006	3	900	2700.00	10/28/03	Shipped	S10_1679	Truck	900	Technics Stores Inc.	6505556809	9408 Furth Circle		Burlingame	CA	94217	USA	NA	Hirano	Juri	Small

2 NF

- It must be in 1NF.
- All non key attributes must be fully dependent on candidate key.
  - i.e. If a non-key column is partially dependent on candidate key (subset of columns forming candidate key), then split them into seperate tables.
- Every table should have primary key and relationship between the tables should be formed using foreign key.

Candidate Key

- It is a set of columns which uniquely identify a record.
- A table can have multiple candidate keys because there can be multiple set of columns which uniquley identify a record/rpw in a table.

Non-key Columns

- Columns which are not part of the candidate key or primary key.

Partial Dependency

- If the candidate key is a combination of two columns (or multiple columns) then every non-key column (columns which are not part of the candudate key) should be fully dependent on all the columns. If there is any non-key column which depends only on one of the candidate key columns, then this results in partial dependency.

2 NF

The table is already in First Normal Form (1NF) (no repeating groups or multi-valued columns).

No partial dependency exists, meaning every non-key column must be fully dependent on the whole primary key, not just part of it.

2NF primarily applies to tables that have composite primary keys (primary keys made of multiple columns).

Student_ID	Student_Name	Course_ID	Course_Name	Instructor	Instructor_Contact
S101	John Doe	C001	Math	Dr. Smith	12345
S101	John Doe	C002	Physics	Dr. Brown	67890
S102	Alice Smith	C001	Math	Dr. Smith	12345

Primary Key: (Student_ID, CourseID)

Here, partial dependency exists

2 NF

Student_ID	Student_Name	Course_ID	Course_Name	Instructor	Instructor_Contact
S101	John Doe	C001	Math	Dr. Smith	12345
S101	John Doe	C002	Physics	Dr. Brown	67890
S102	Alice Smith	C001	Math	Dr. Smith	12345

Student_ID	Student_Name
S101	John Doe
S102	Alice Smith

Course_ID	Course_Name	Instructor	Instructor_Contact
C001	Math	Dr. Smith	12345
C002	Physics	Dr. Brown	67890

Fix the relationship for two splitted tables

Student_ID	Course_ID
S101	C001
S101	C002
S102	C001

Many to Many Relationship

2 NF

It must be in 1NF.
All non key attributes must be fully dependent on candidate key.
i.e. If a non-key column is partially dependent on candidate key (subset of columns forming candidate key), then split them into seperate tables.
Every table should have primary key and relationship between the tables should be formed using foreign key.

Candidate Key

It is a set of columns which uniquely identify a record.
A table can have multiple candidate keys because there can be multiple set of columns which uniquley identify a record/rpw in a table.

Non-key Columns

Columns which are not part of the candidate key or primary key.

Partial Dependency

If the candidate key is a combination of two columns (or multiple columns) then every non-key column (columns which are not part of the candudate key) should be fully dependent on all the columns. If there is any non-key column which depends only on one of the candidate key columns, then this results in partial dependency.

ORDER_NUMBER	QUANTITY_ORDERED	PRICE_EACH	SALES	ORDER_DATE	STATUS	PRODUCT_CODE	PRODUCT_NAME	MSRP	CUSTOMER_NAME	PHONE	ADDRESS_LINE1	ADDRESS_LINE2	CITY	STATE	POSTAL_CODE	COUNTRY	TERRITORY	CONTACT_LAST_NAME	CONTACT_FIRST_NAME	DEAL_SIZE
1001	30	95.7	2871.00	2/24/03	Shipped	S10_1678	Motorcycles	95	Land of Toys	2125557818	897 Long Airport Avenue		NYC	NY	10022	USA	NA	Yu	Kwai	Small
1002	34	81.35	2765.90	5/7/03	Shipped	S10_1678	Motorcycles	95	Reims Collec	26.47.1555	59 rue de l'Abbaye		Reims		51100	France	EMEA	Henriot	Paul	Small
1003	41	94.74	3884.34	7/1/03	Shipped	S10_1678	Motorcycles	95	Lyon Souven	1 46 62 7555	27 rue du Colonel Pierre Av		Paris		75508	France	EMEA	Da Cunha	Daniel	Medium
1004	45	83.26	3746.70	8/25/03	Shipped	S10_1678	Motorcycles	95	Toys4Grownu	6265557265	78934 Hillside Dr.		Pasadena	CA	90003	USA	NA	Young	Julie	Medium
1005	49	100	5205.27	10/10/03	Shipped	S10_1678	Motorcycles	95	Corporate Gi	6505551386	7734 Strong St.		San Francisc	CA		USA	NA	Brown	Julie	Medium
1006	4	800.5	3202.00	10/28/03	Shipped	S10_1678	Car	800	Technics Sto	6505556809	9408 Furth Circle		Burlingame	CA	94217	USA	NA	Hirano	Juri	Small
1006	3	900	2700.00	10/28/03	Shipped	S10_1679	Truck	900	Technics Sto	6505556809	9408 Furth Circle		Burlingame	CA	94217	USA	NA	Hirano	Juri	Small

Candidate key = ORDER_NUMBER || PRODUCT_CODE
Non-key columns are partially dependent on candidate key

1. QUANTITY_ORDERED, PRICE_EACH, SALES, ORDER_DATE, STATUS, CONTACT_LAST_NAME, CONTACT_FIRST_NAME, DEAL_SIZE are dependent only on OREDER_NUMBER
2. PRODUCT_NAME, MSRP are dependent only on PRODUCT_CODE
3. PHONE, ADDRESS_LINE1, ADDRESS_LINE2, CITY, STATE, POSTAL_CODE, COUNTRY, TERRITORY are neither dependent on OREDER_NUMBER nor PRODUCT_CODE. But they are dependent on CUSTOMER_NAME.

2. Products

3. Customers

ORDER_NUMBER	QUANTITY_ORDERED	PRICE_EACH	SALES	ORDER_DATE	STATUS	PRODUCT_CODE	PRODUCT_NAME	MSRP	CUSTOMER_NAME	PHONE	ADDRESS_LINE1	ADDRESS_LINE2	CITY	STATE	POSTAL_CODE	COUNTRY	TERRITORY	CONTACT_LAST_NAME	CONTACT_FIRST_NAME	DEAL_SIZE
1001	30	95.7	2871.00	2/24/03	Shipped	S10_1678	Motorcycles	95	Land of Toys	2125557818	897 Long Airport Avenue		NYC	NY	10022	USA	NA	Yu	Kwai	Small
1002	34	81.35	2765.90	5/7/03	Shipped	S10_1678	Motorcycles	95	Reims Collec	26.47.1555	59 rue de l'Abbaye		Reims		51100	France	EMEA	Henriot	Paul	Small
1003	41	94.74	3884.34	7/1/03	Shipped	S10_1678	Motorcycles	95	Lyon Souven	1 46 62 7555	27 rue du Colonel Pierre A		Paris		75508	France	EMEA	Da Cunha	Daniel	Medium
1004	45	83.26	3746.70	8/25/03	Shipped	S10_1678	Motorcycles	95	Toys4Grownl	6265557265	78934 Hillside Dr.		Pasadena	CA	90003	USA	NA	Young	Julie	Medium
1005	49	100	5205.27	10/10/03	Shipped	S10_1678	Motorcycles	95	Corporate Gi	6505551386	7734 Strong St.		San Francisc	CA		USA	NA	Brown	Julie	Medium
1006	4	800.5	3202.00	10/28/03	Shipped	S10_1678	Car	800	Technics Sto	6505556809	9408 Furth Circle		Burlingame	CA	94217	USA	NA	Hirano	Juri	Small
1006	3	900	2700.00	10/28/03	Shipped	S10_1679	Truck	900	Technics Sto	6505556809	9408 Furth Circle		Burlingame	CA	94217	USA	NA	Hirano	Juri	Small

1. Orders

1. Orders

Create 3 tables

PRODUCTS

PRODUCT_CODE	PRODUCT_NAME	MSRP
S10_1678	Motorcycles	95
S10_1678	Motorcycles	95
S10_1678	Motorcycles	95
S10_1678	Motorcycles	95
S10_1678	Motorcycles	95
S10_1678	Car	800
S10_1679	Truck	900

CUSTOMERS

CUSTOMER_NAME	PHONE	ADDRESS_LINE1	ADDRESS_LINE2	CITY	STATE	POSTAL_CODE	COUNTRY	TERRITORY
Land of Toys	2125557818	897 Long Airport Avenue		NYC	NY	10022	USA	NA
Reims Collec	26.47.1555	59 rue de l'Abbaye		Reims		51100	France	EMEA
Lyon Souveni	1 46 62 7555	27 rue du Colonel Pierre Av		Paris		75508	France	EMEA
Toys4Grownu	6265557265	78934 Hillside Dr.		Pasadena	CA	90003	USA	NA
Corporate Gi	6505551386	7734 Strong St.		San Franc	CA		USA	NA
Technics Sto	6505556809	9408 Furth Circle		Burlingam	CA	94217	USA	NA
Technics Sto	6505556809	9408 Furth Circle		Burlingam	CA	94217	USA	NA

ORDERS

ORDER_NUMBER	QUANTITY_ORDERED	PRICE_EACH	SALES	ORDER_DATE	STATUS	CONTACT_LAST_NAME	CONTACT_FIRST_NAME	DEAL_SIZE
1001	30	95.7	2871.00	2/24/03	Shipped	Yu	Kwai	Small
1002	34	81.35	2765.90	5/7/03	Shipped	Henriot	Paul	Small
1003	41	94.74	3884.34	7/1/03	Shipped	Da Cunha	Daniel	Medium
1004	45	83.26	3746.70	8/25/03	Shipped	Young	Julie	Medium
1005	49	100	5205.27	10/10/03	Shipped	Brown	Julie	Medium
1006	4	800.5	3202.00	10/28/03	Shipped	Hirano	Juri	Small
1006	3	900	2700.00	10/28/03	Shipped	Hirano	Juri	Small

Every table should have primary key and relationship between the tables should be formed using foreign key.

First, fix primary key

PRODUCTS

PRODUCT_CODE	PRODUCT_NAME	MSRP
S10_1678	Motorcycles	95
S10_1678	Motorcycles	95
S10_1678	Motorcycles	95
S10_1678	Motorcycles	95
S10_1678	Motorcycles	95
S10_1678	Car	800
S10_1679	Truck	900

>>

PRODUCT_CODE	PRODUCT_NAME	MSRP
S10_1678	Motorcycles	95
S10_1678	Car	800
S10_1679	Truck	900

CUSTOMERS

CUSTOMER_NAME	PHONE	ADDRESS_LINE1	ADDRESS_LINE2	CITY	STATE	POSTAL_CODE	COUNTRY	TERRITORY
Land of Toys	2125557818	897 Long Airport Avenue		NYC	NY	10022	USA	NA
Reims Collec	26.47.1555	59 rue de l'Abbaye		Reims		51100	France	EMEA
Lyon Souveni	1 46 62 7555	27 rue du Colonel Pierre Av		Paris		75508	France	EMEA
Toys4Grownl	6265557265	78934 Hillside Dr.		Pasadena	CA	90003	USA	NA
Corporate Gi	6505551386	7734 Strong St.		San Franc	CA		USA	NA
Technics Sto	6505556809	9408 Furth Circle		Burlingam	CA	94217	USA	NA
Technics Sto	6505556809	9408 Furth Circle		Burlingam	CA	94217	USA	NA

>>

CUSTOMERS

CUSTMER_ID	CUSTOMER_NAME	PHONE	ADDRESS_LINE1	ADDRESS_LINE2	CITY	STATE	POSTAL_CODE	COUNTRY	TERRITORY
1	Land of Toys	2.126E+09	897 Long Airport Avenue		NYC	NY	10022	USA	NA
2	Reims Collec	26.47.1555	59 rue de l'Abbaye		Reims		51100	France	EMEA
3	Lyon Souven	1 46 62 7555	27 rue du Colonel Pierre Av		Paris		75508	France	EMEA
4	Toys4Grownl	6.266E+09	78934 Hillside Dr.		Pasadena	CA	90003	USA	NA
5	Corporate Gi	6.506E+09	7734 Strong St.		San Francisc	CA		USA	NA
6	Technics Sto	6.506E+09	9408 Furth Circle		Burlingame	CA	94217	USA	NA
7	Technics Sto	6.506E+09	9408 Furth Circle		Burlingame	CA	94217	USA	NA

ORDERS

ORDER_NUMBER	QUANTITY_ORDERED	PRICE_EACH	SALES	ORDER_DATE	STATUS	CONTACT_LAST_NAME	CONTACT_FIRST_NAME	DEAL_SIZE
1001	30	95.7	2871.00	2/24/03	Shipped	Yu	Kwai	Small
1002	34	81.35	2765.90	5/7/03	Shipped	Henriot	Paul	Small
1003	41	94.74	3884.34	7/1/03	Shipped	Da Cunha	Daniel	Medium
1004	45	83.26	3746.70	8/25/03	Shipped	Young	Julie	Medium
1005	49	100	5205.27	10/10/03	Shipped	Brown	Julie	Medium
1006	4	800.5	3202.00	10/28/03	Shipped	Hirano	Juri	Small
1006	3	900	2700.00	10/28/03	Shipped	Hirano	Juri	Small

>>

ORDERS

ORDER_ID	ORDER_NUMBER	QUANTITY_ORDERED	PRICE_EACH	SALES	ORDER_DATE	STATUS	CONTACT_LAST_NAME	CONTACT_FIRST_NAME	DEAL_SIZE
1	1001	30	95.7	2871.00	2/24/03	Shipped	Yu	Kwai	Small
2	1002	34	81.35	2765.90	5/7/03	Shipped	Henriot	Paul	Small
3	1003	41	94.74	3884.34	7/1/03	Shipped	Da Cunha	Daniel	Medium
4	1004	45	83.26	3746.70	8/25/03	Shipped	Young	Julie	Medium
5	1005	49	100	5205.27	10/10/03	Shipped	Brown	Julie	Medium
6	1006	4	800.5	3202.00	10/28/03	Shipped	Hirano	Juri	Small
7	1006	3	900	2700.00	10/28/03	Shipped	Hirano	Juri	Small

Second, fix foreign keys

ORDERS

ORDER_ID	ORDER_NUMBER	QUANTITY_ORDERED	PRICE_EACH	SALES	ORDER_DATE	STATUS	CONTACT_LAST_NAME	CONTACT_FIRST_NAME	DEAL_SIZE	PRODUCT_CODE	CUSTMER_ID
1	1001	30	95.7	2871.00	2/24/03	Shipped	Yu	Kwai	Small	S10_1678	1
2	1002	34	81.35	2765.90	5/7/03	Shipped	Henriot	Paul	Small	S10_1678	2
3	1003	41	94.74	3884.34	7/1/03	Shipped	Da Cunha	Daniel	Medium	S10_1678	3
4	1004	45	83.26	3746.70	8/25/03	Shipped	Young	Julie	Medium	S10_1678	4
5	1005	49	100	5205.27	10/10/03	Shipped	Brown	Julie	Medium	S10_1678	5
6	1006	4	800.5	3202.00	10/28/03	Shipped	Hirano	Juri	Medium	S10_1678	6
7	1006	3	900	2700.00	10/28/03	Shipped	Hirano	Juri	Medium	S10_1679	7

OR

RELATIONSHIPS

ORDER_ID	PRODUCT_CODE	CUSTMER_ID
1	S10_1678	1
2	S10_1678	2
3	S10_1678	3
4	S10_1678	4
5	S10_1678	5
6	S10_1678	6
7	S10_1679	7

3 NF

- The table is already in Second Normal Form (2NF) (no partial dependencies).
- No transitive dependency exists, meaning all non-key attributes depend only on the primary key and not on another non-key attribute.
  - Consider a table, T which has 3 columns namely A, B and C.
  - If A is functionally dependent on B and B is functionally dependent on C, then we can say that A is functionally dependent on C.

3 NF

Student_ID	Student_Name
S101	John Doe
S102	Alice Smith

✅

Course_ID	Course_Name	Instructor	Instructor_Contact
C001	Math	Dr. Smith	12345
C002	Physics	Dr. Brown	67890

❌

Student_ID	Course_ID
S101	C001
S101	C002
S102	C001

✅

3 NF

Course_ID	Course_Name	Instructor	Instructor_Contact
C001	Math	Dr. Smith	12345
C002	Physics	Dr. Brown	67890

X

Course_ID	Course_Name	Instructor_Id
C001	Math	I101
C002	Physics	I102

✓

Instructor_Id	Instructor	Instructor_Contact
I101	Dr. Smith	12345
I102	Dr. Brown	67890

✓

PRODUCTS

PRODUCT_CODE	PRODUCT_NAME	MSRP
S10_1678	Motorcycles	95
S10_1678	Car	800
S10_1679	Truck	900

CUSTOMERS

CUSTMER_ID	CUSTOMER_NAME	PHONE	ADDRESS_LINE1	ADDRESS_LINE2	CITY	STATE	POSTAL_CODE	COUNTRY	TERRITORY
1	Land of Toys I	2125557818	897 Long Airport Avenue		NYC	NY	10022	USA	NA
2	Reims Collec	26.47.1555	59 rue de l'Abbaye		Reims		51100	France	EMEA
3	Lyon Souveni	1 46 62 7555	27 rue du Colonel Pierre Av		Paris		75508	France	EMEA
4	Toys4Grownl	6265557265	78934 Hillside Dr.		Pasadena	CA	90003	USA	NA
5	Corporate Gi	6505551386	7734 Strong St.		San Francisc	CA		USA	NA
6	Technics Sto	6505556809	9408 Furth Circle		Burlingame	CA	94217	USA	NA
7	Technics Sto	6505556809	9408 Furth Circle		Burlingame	CA	94217	USA	NA

ORDERS

ORDER_ID	ORDER_NUMBER	QUANTITY_ORDERED	PRICE_EACH	SALES	ORDER_DATE	STATUS	CONTACT_LAST_NAME	CONTACT_FIRST_NAME	DEAL_SIZE
1	1001	30	95.7	2871.00	2/24/03	Shipped	Yu	Kwai	Small
2	1002	34	81.35	2765.90	5/7/03	Shipped	Henriot	Paul	Small
3	1003	41	94.74	3884.34	7/1/03	Shipped	Da Cunha	Daniel	Medium
4	1004	45	83.26	3746.70	8/25/03	Shipped	Young	Julie	Medium
5	1005	49	100	5205.27	10/10/03	Shipped	Brown	Julie	Medium
6	1006	4	800.5	3202.00	10/28/03	Shipped	Hirano	Juri	Small
7	1006	3	900	2700.00	10/28/03	Shipped	Hirano	Juri	Small

C

RELATIONSHIPS

ORDER_ID	PRODUCT_CODE	CUSTMER_ID
1	S10_1678	1
2	S10_1678	2
3	S10_1678	3
4	S10_1678	4
5	S10_1678	5
6	S10_1678	6
7	S10_1679	7

B A

Here Orders and Customers violate 3NF due to transitive dependency.

Solution

ORDERS

ORDER_ID	ORDER_NUMBER	QUANTITY_ORDERED	PRICE_EACH	SALES	ORDER_DATE	STATUS	DEAL_SIZE
1	1001	30	95.7	2871.00	2/24/03	Shipped	Small
2	1002	34	81.35	2765.90	5/7/03	Shipped	Small
3	1003	41	94.74	3884.34	7/1/03	Shipped	Medium
4	1004	45	83.26	3746.70	8/25/03	Shipped	Medium
5	1005	49	100	5205.27	10/10/03	Shipped	Medium
6	1006	4	800.5	3202.00	10/28/03	Shipped	Small
7	1006	3	900	2700.00	10/28/03	Shipped	Small

EMPLOYEES

EMP_ID	CONTACT_LAST_NAME	CONTACT_FIRST_NAME
1	Yu	Kwai
2	Henriot	Paul
3	Da Cunha	Daniel
4	Young	Julie
5	Brown	Julie
6	Hirano	Juri

RELATIONSHIPS

ORDER_ID	PRODUCT_CODE	CUSTMER_ID	EMP_ID
1	S10_1678	1	1
2	S10_1678	2	2
3	S10_1678	3	3
4	S10_1678	4	4
5	S10_1678	5	5
6	S10_1678	6	6
7	S10_1679	7	6

CUSTOMERS

CUSTOMER_ID (PK)	CUSTOMER_NAME	PHONE	ADDRESS_LINE1	ADDRESS_LINE2	POSTAL_CODE (FK)
1	Land of Toys Inc.	2125557818	897 Long Airport Avenue		10022
2	Reims Collectables	26.47.1555	59 rue de l'Abbaye		51100
3	Lyon Souveniers	1 46 62 7555	27 rue du Colonel Pierre Avia		75508
4	Toys4GrownUps.cor	6265557265	78934 Hillside Dr.		90003
5	Corporate Gift Ideas	6505551386	7734 Strong St.		
6	Technics Stores Inc.	6505556809	9408 Furth Circle		94217
7	Technics Stores Inc.	6505556809	9408 Furth Circle		94217

POSTAL_CODE (PK(	CITY	STATE	COUNTRY_ID (FK)
10022	NYC	NY	1
51100	Reims		2
75508	Paris		2
90003	Pasadena	CA	1
	San Francisco	CA	1
94217	Burlingame	CA	1
94217	Burlingame	CA	1

COUNTRY_ID (PK)	COUNTRY	TERRITORY (FK)
1	USA	1
2	France	2

TERRITORY_ID (PK)	TERRITORY
1	NA
2	EMEA

Connecting the Parts of Knowledge With the Wholeness of Knowledge

UNITY CHART

CONNECTING THE PARTS OF KNOWLEDGE WITH THE WHOLENESS OF KNOWLEDGE:

Relational Model Basics & Relational Algebra Operations

1. Science repeatedly calls upon mathematics to help it model the real world.
2. The operators of the relational algebra have been found to be ideal for modeling the operations that are needed to query and update a database.

---

3. **Transcendental consciousness** is the experience of the simplest and most abstract state of awareness which underlies all states of greater excitation.
4. **Impulses within the Transcendental Field**: Nature accomplishes what it needs by having its impulses in the transcendental field be as efficient as possible.
5. **Wholeness moving within itself**: In unity consciousness one experiences everything as excitations of pure consciousness that underlies and connects all diversity.