SQL Workshop

outline

0] WHAT IS SQL?

1] WHAT IS PROGRAMMING LANGUAGE?

2] TYPE OF PROGRAMMING LANGUAGE.

3] WHAT IS A DATABASE?

4] STRUCTURED & UNSTRUCTURED DATA

5] RELATIONAL & NON-RELATIONAL

6] SQL (WHAT CAN SQL DO?)

7] TABLE META DATA

8] SCHEMA

9] ENTITIES – ATTRIBUTES - RELATIONSHIPS

8] TYPE OF RELATIONAL DB (+EXAMPLE).

9] SQL CODES

Eng Dana Mohammed Al-Mahrouk

- Front-end Developer | Database administrator

Bachelor’s degree in Computer Engineering from BAU.

- Two years of experience in databases.
- Course in databases.
- Front-end projects.

LinkedIn link:

almahrouk6103@gmail.com

What is SQL?

Structured query language (SQL) is a programming language for storing and processing information in a relational database.

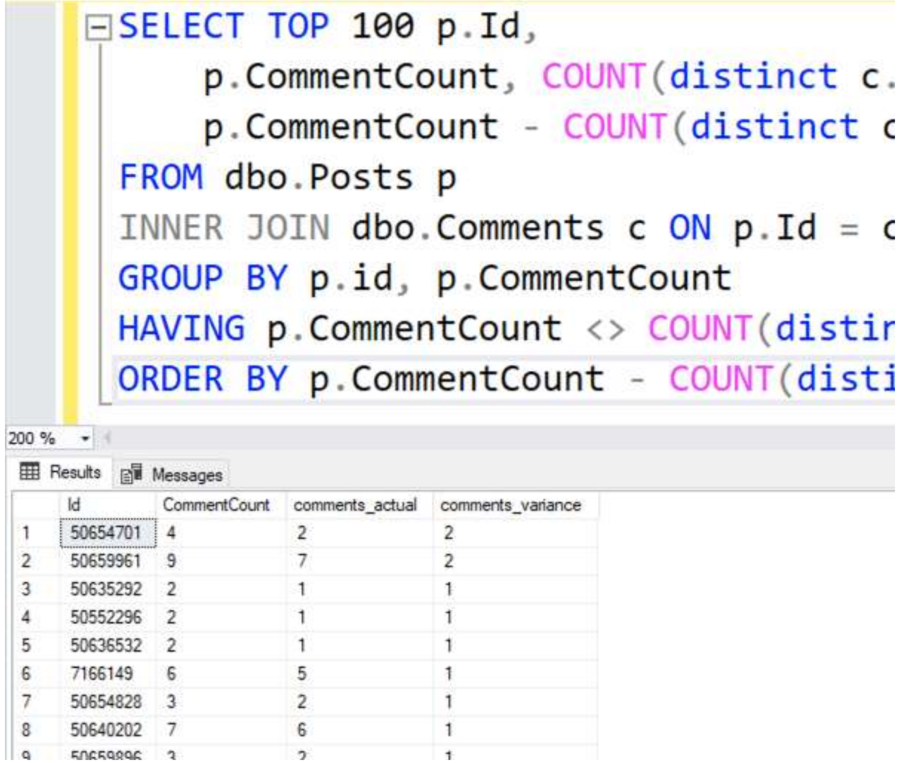
A relational database stores information in table form, with rows and columns representing different data attributes and the various relationships between the data values.

What is programming language ?

A programming language is a formal set of **instructions** that are used to communicate with computers.

write code that can be understood by computers and executed to perform various tasks or solve specific problems.

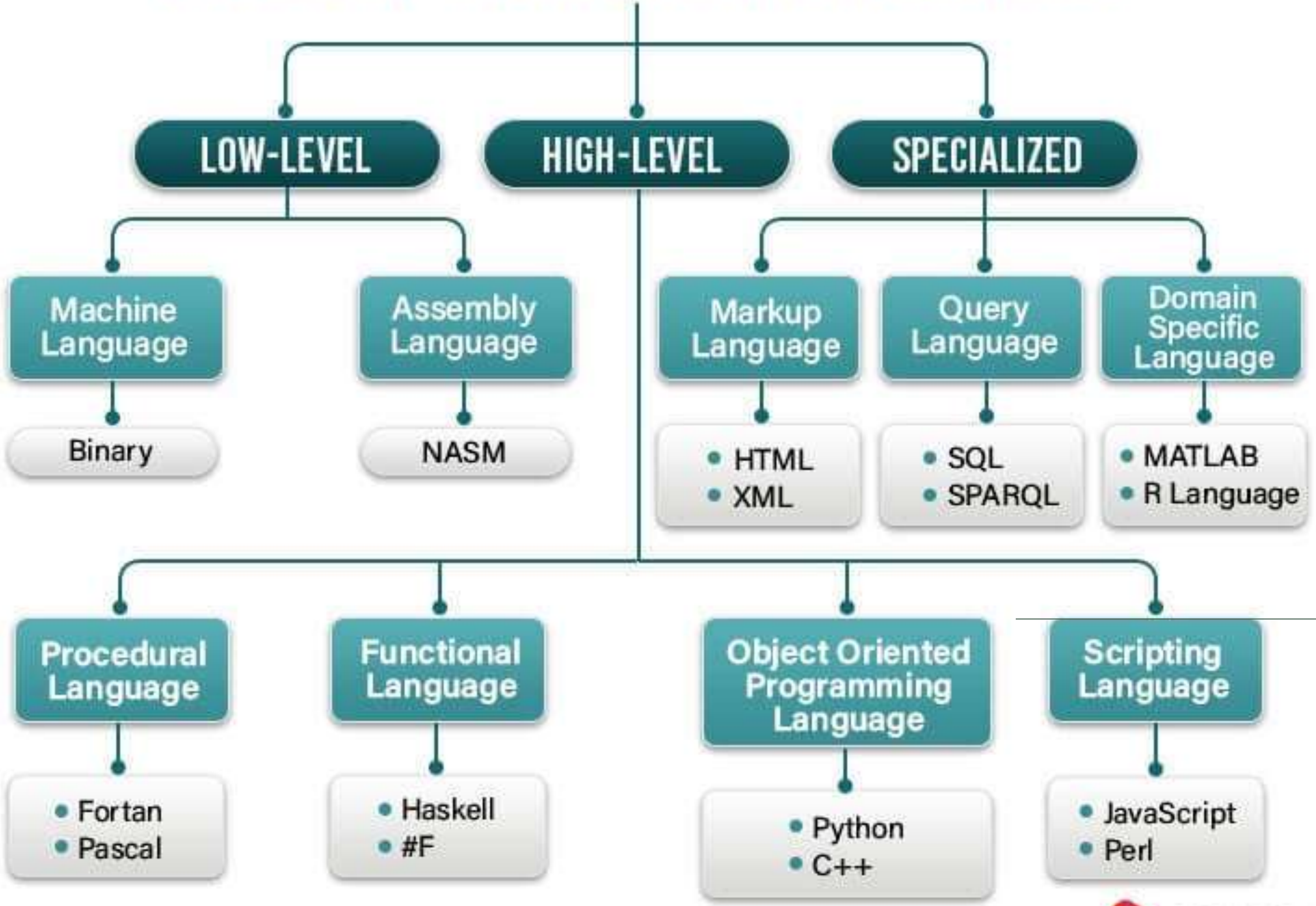
Programming languages vary in complexity, purpose, and application. They can be used for a wide range of tasks, including web development, software development, data analysis, artificial intelligence, and more.



```sql
SELECT TOP 100 p.Id,
    p.CommentCount, COUNT(distinct c.
    p.CommentCount - COUNT(distinct c
FROM dbo.Posts p
INNER JOIN dbo.Comments c ON p.Id = c
GROUP BY p.id, p.CommentCount
HAVING p.CommentCount <> COUNT(distir
ORDER BY p.CommentCount - COUNT(disti
```

	Id	CommentCount	comments_actual	comments_variance
1	50654701	4	2	2
2	50659961	9	7	2
3	50635292	2	1	1
4	50552296	2	1	1
5	50636532	2	1	1
6	7166149	6	5	1
7	50654828	3	2	1
8	50640202	7	6	1
9	50659896	3	2	1

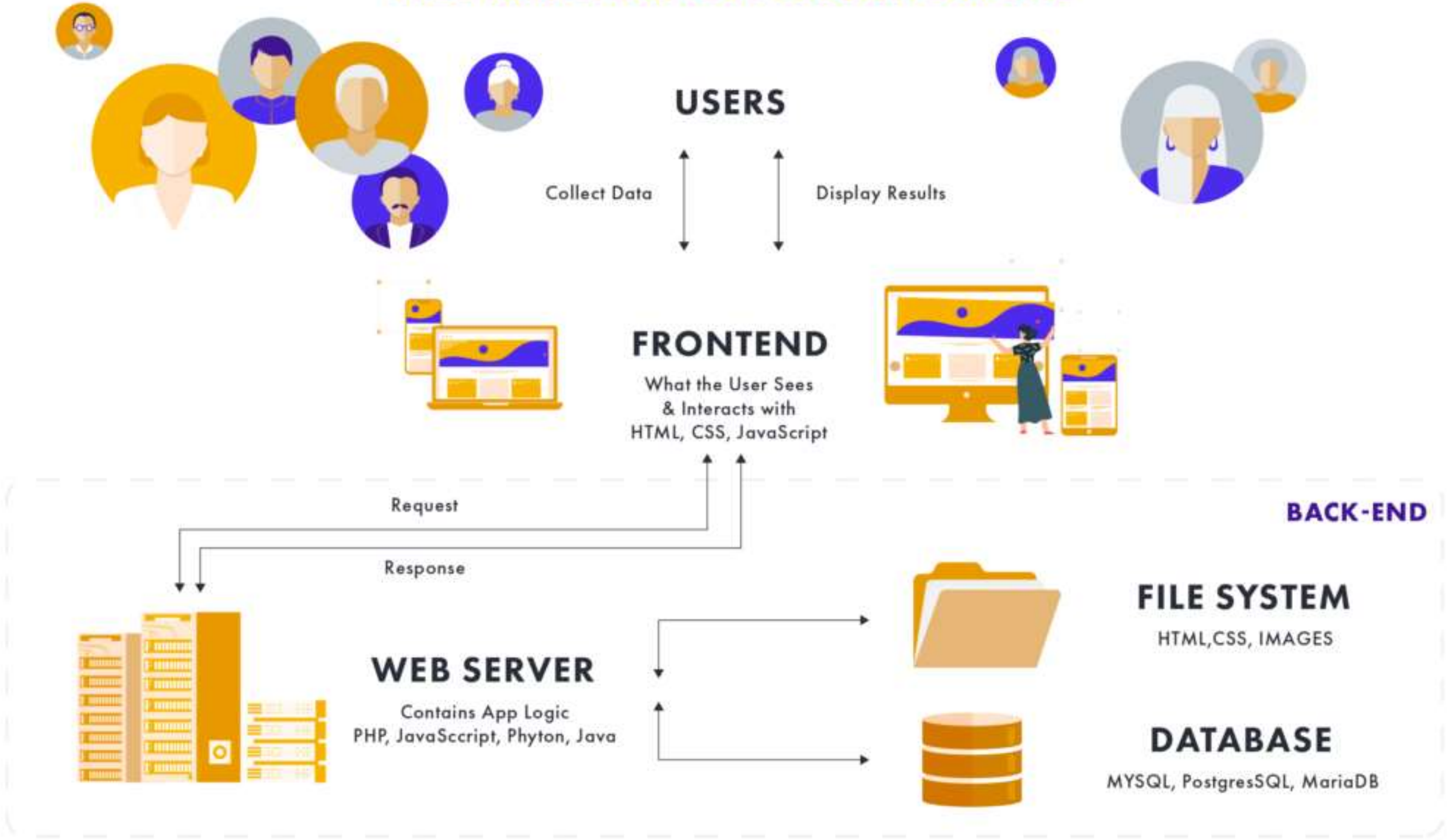
TYPES OF COMPUTER LANGUAGES



- LOW-LEVEL
  - Machine Language
    - Binary
  - Assembly Language
    - NASM
- HIGH-LEVEL
  - Procedural Language
    - Fortan
    - Pascal
  - Functional Language
    - Haskell
    - #F
  - Object Oriented Programming Language
    - Python
    - C++
  - Scripting Language
    - JavaScript
    - Perl
- SPECIALIZED
  - Markup Language
    - HTML
    - XML
  - Query Language
    - SQL
    - SPARQL
  - Domain Specific Language
    - MATLAB
    - R Language

EDUCBA

WEB APPLICATION ARCHITECTURE



USERS

Collect Data

Display Results

FRONTEND

What the User Sees
& Interacts with
HTML, CSS, JavaScript

Request

Response

BACK-END

WEB SERVER

Contains App Logic
PHP, JavaSccript, Phyton, Java

FILE SYSTEM

HTML,CSS, IMAGES

DATABASE

MYSQL, PostgresSQL, MariaDB

What is a Database?

A **Database** is a ***structured collection of data*** that is **organized and stored** in a way that enables efficient retrieval, updating, and management of that data.

It serves as a Store of information that can be easily accessed, managed, and manipulated by users or applications.

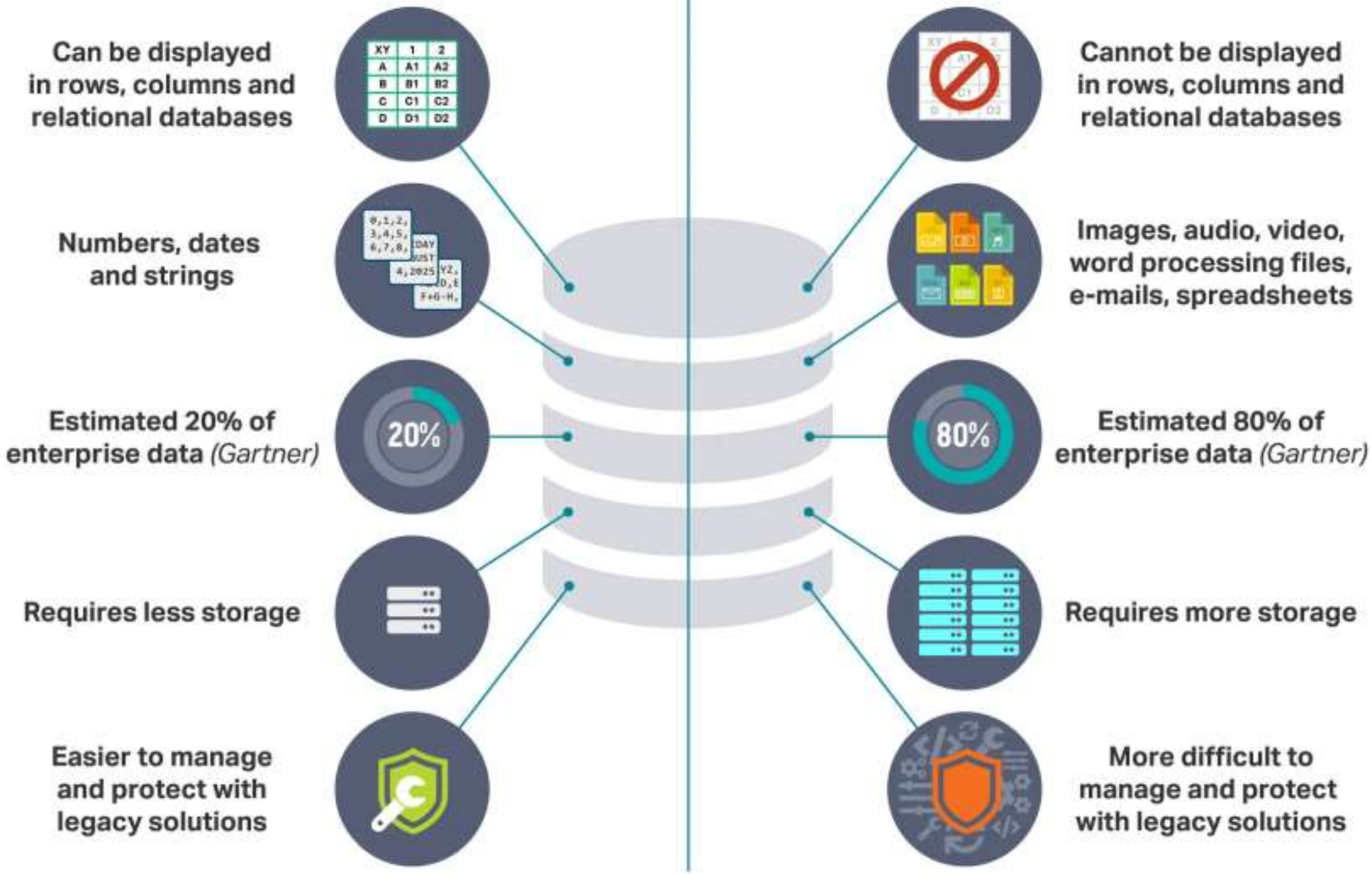
- commonly used:

business, science, academia, and government.

- Types:

relational DB, NoSQL, object-oriented DB, …

Structured Data vs Unstructured Data

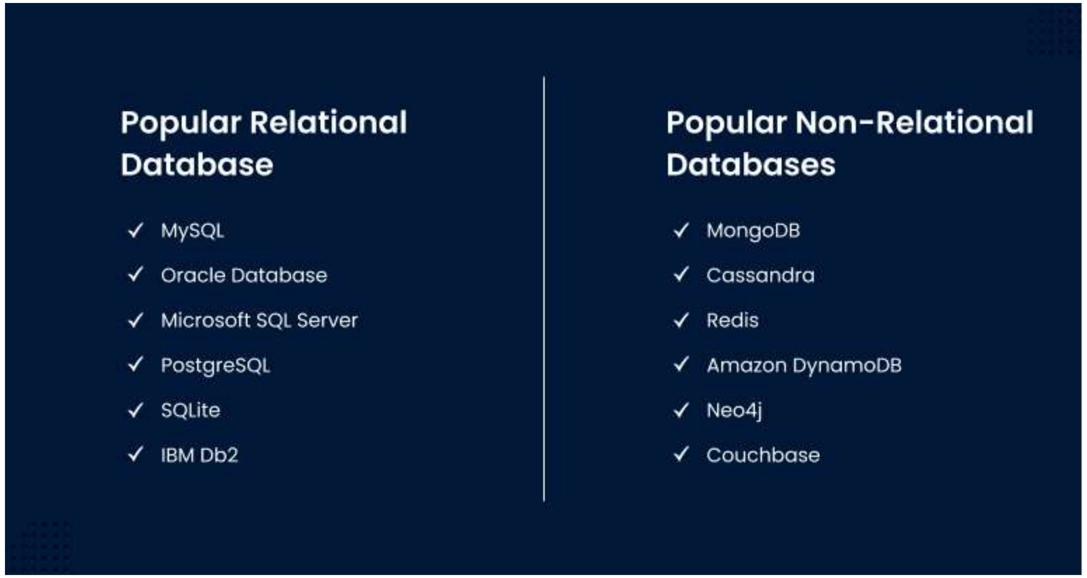


Structured Data

- Can be displayed in rows, columns and relational databases
- Numbers, dates and strings
- Estimated 20% of enterprise data *(Gartner)*
- Requires less storage
- Easier to manage and protect with legacy solutions

Unstructured Data

- Cannot be displayed in rows, columns and relational databases
- Images, audio, video, word processing files, e-mails, spreadsheets
- Estimated 80% of enterprise data *(Gartner)*
- Requires more storage
- More difficult to manage and protect with legacy solutions

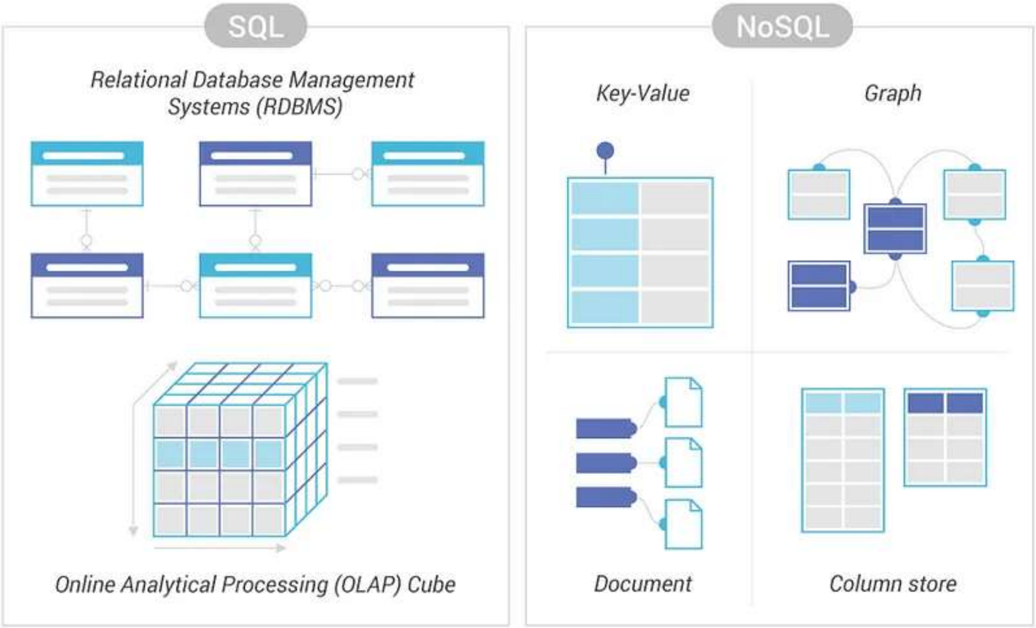


Popular Relational Database

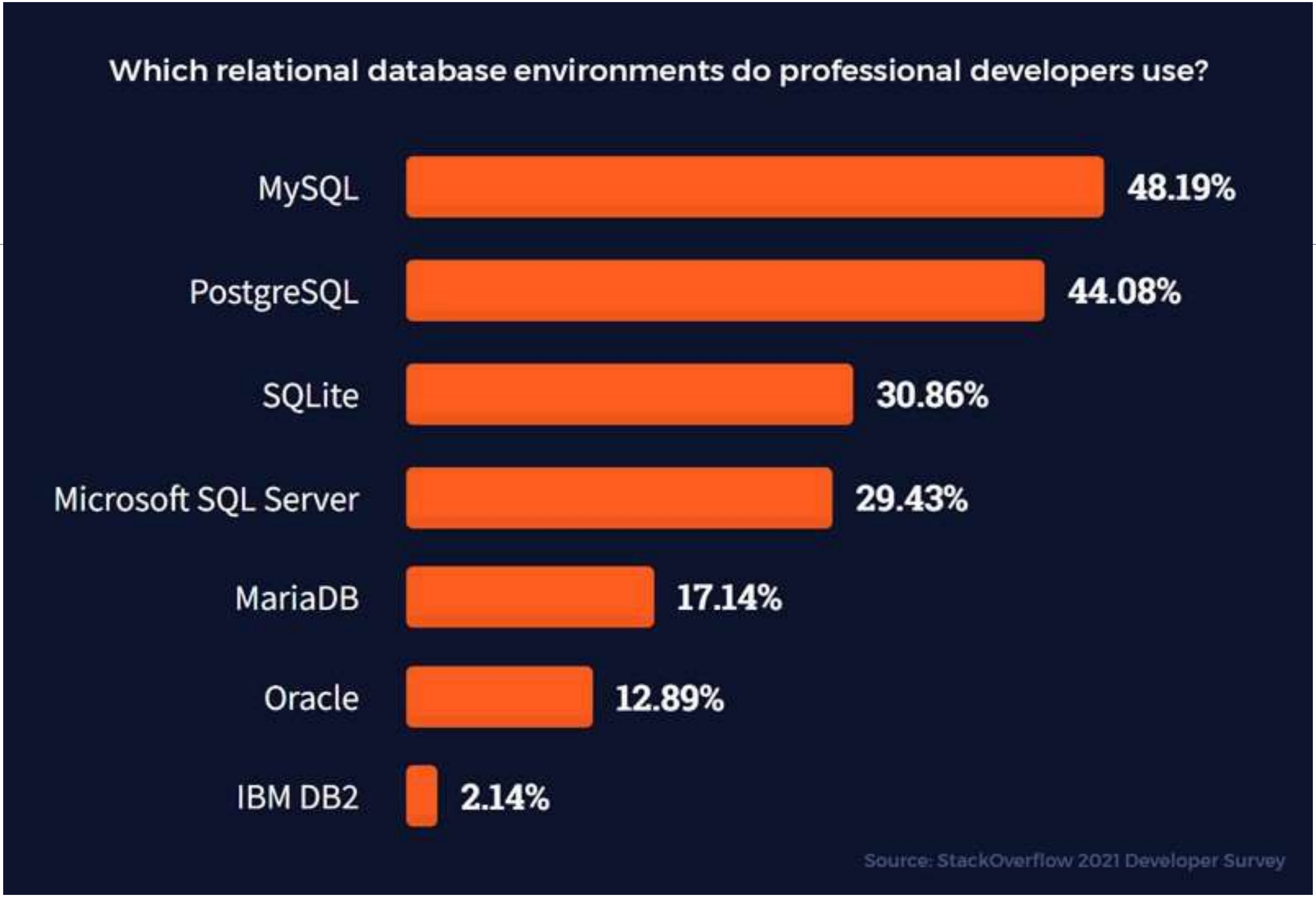
- ✓ MySQL
- ✓ Oracle Database
- ✓ Microsoft SQL Server
- ✓ PostgreSQL
- ✓ SQLite
- ✓ IBM Db2

Popular Non-Relational Databases

- ✓ MongoDB
- ✓ Cassandra
- ✓ Redis
- ✓ Amazon DynamoDB
- ✓ Neo4j
- ✓ Couchbase


SQL
Relational Database Management Systems (RDBMS)
Online Analytical Processing (OLAP) Cube
NoSQL
Key-Value
Graph
Document
Column store

Relational vs Non-relational



Source: StackOverflow 2021 Developer Survey

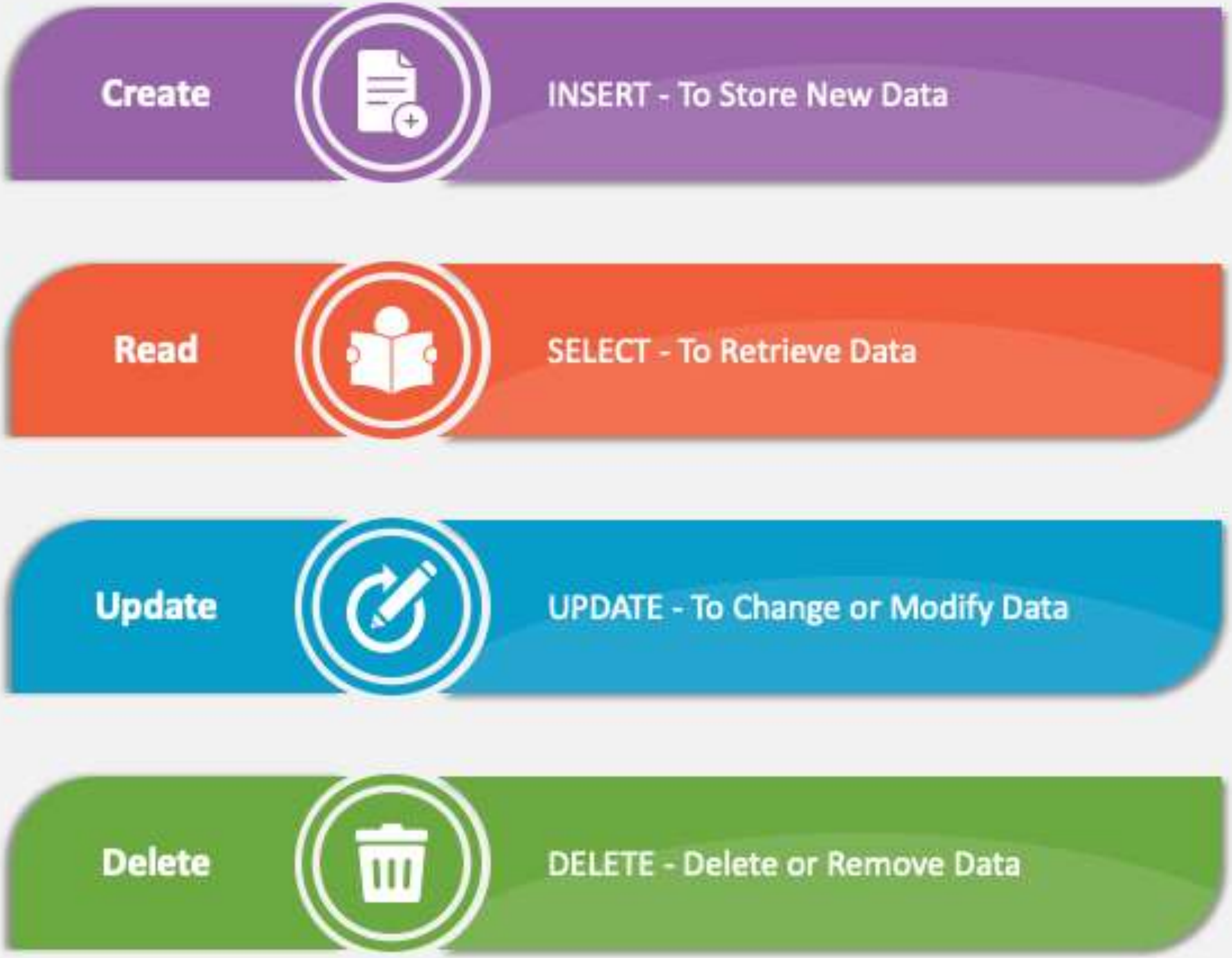
SQL

What Can SQL do?

- execute **queries** against a database
- **insert** records in a database
- **read** data from a database
- **update** records in a database
- **delete** records from a database

- create new **databases**
- create new **tables** in a database
- create stored procedures in a database
- create **views** in a database
- set **permissions** on tables, procedures, and views

CREATE READ UPDATE DELETE



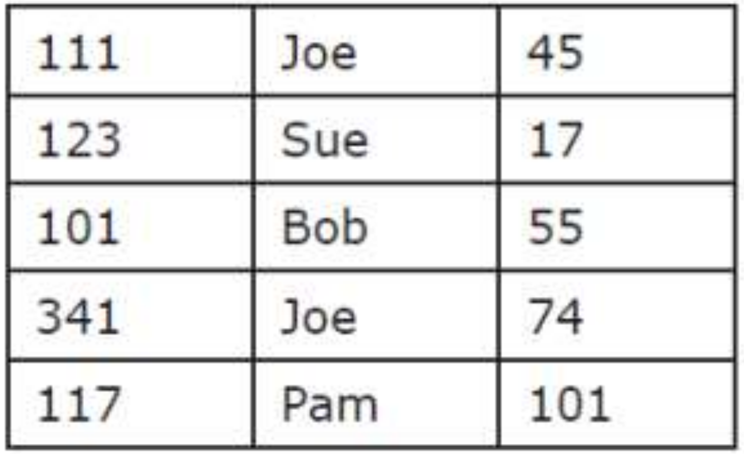
Create — INSERT - To Store New Data

Read — SELECT - To Retrieve Data

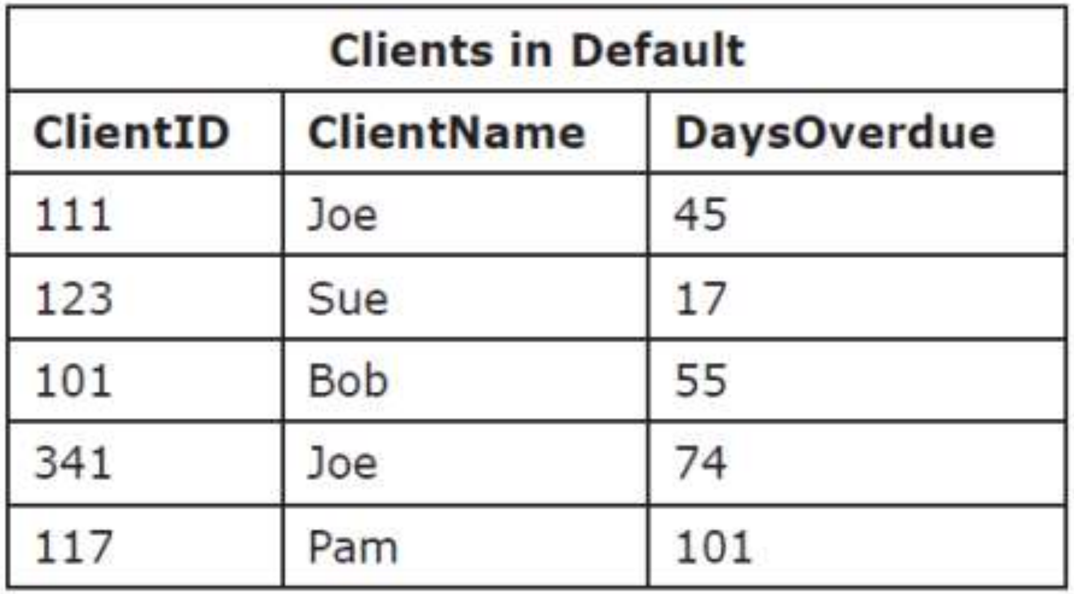
Update — UPDATE - To Change or Modify Data

Delete — DELETE - Delete or Remove Data

Table Meta Data

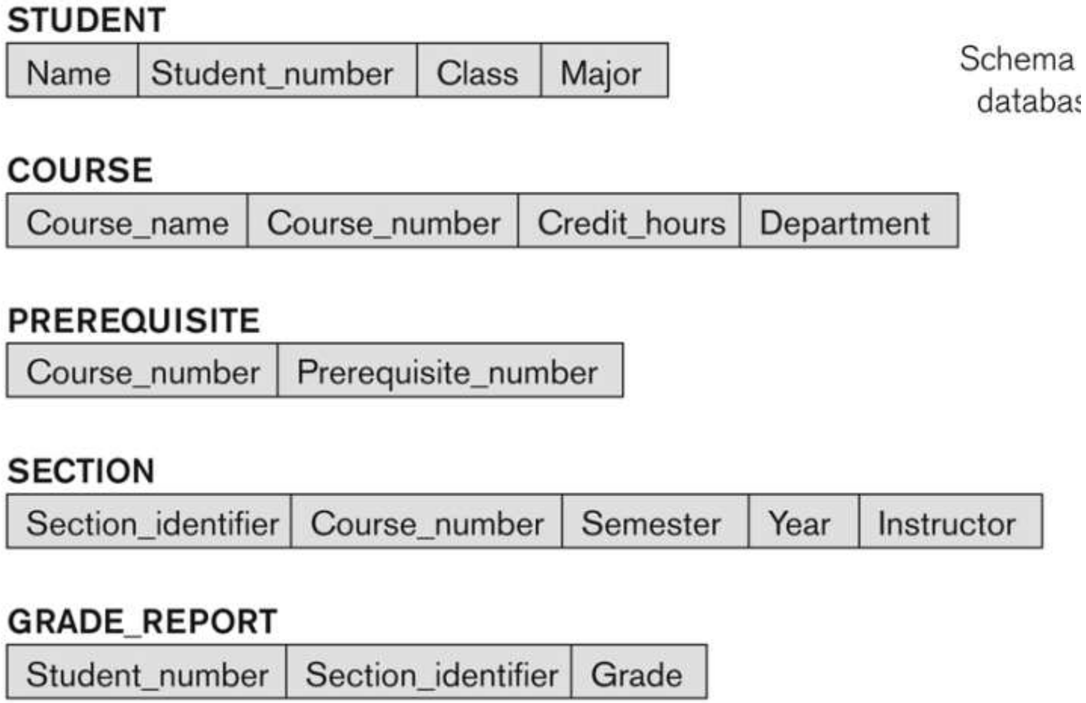


111	Joe	45
123	Sue	17
101	Bob	55
341	Joe	74
117	Pam	101



Clients in Default		
ClientID	**ClientName**	**DaysOverdue**
111	Joe	45
123	Sue	17
101	Bob	55
341	Joe	74
117	Pam	101

Schema



STUDENT

Name	Student_number	Class	Major

COURSE

Course_name	Course_number	Credit_hours	Department

PREREQUISITE

Course_number	Prerequisite_number

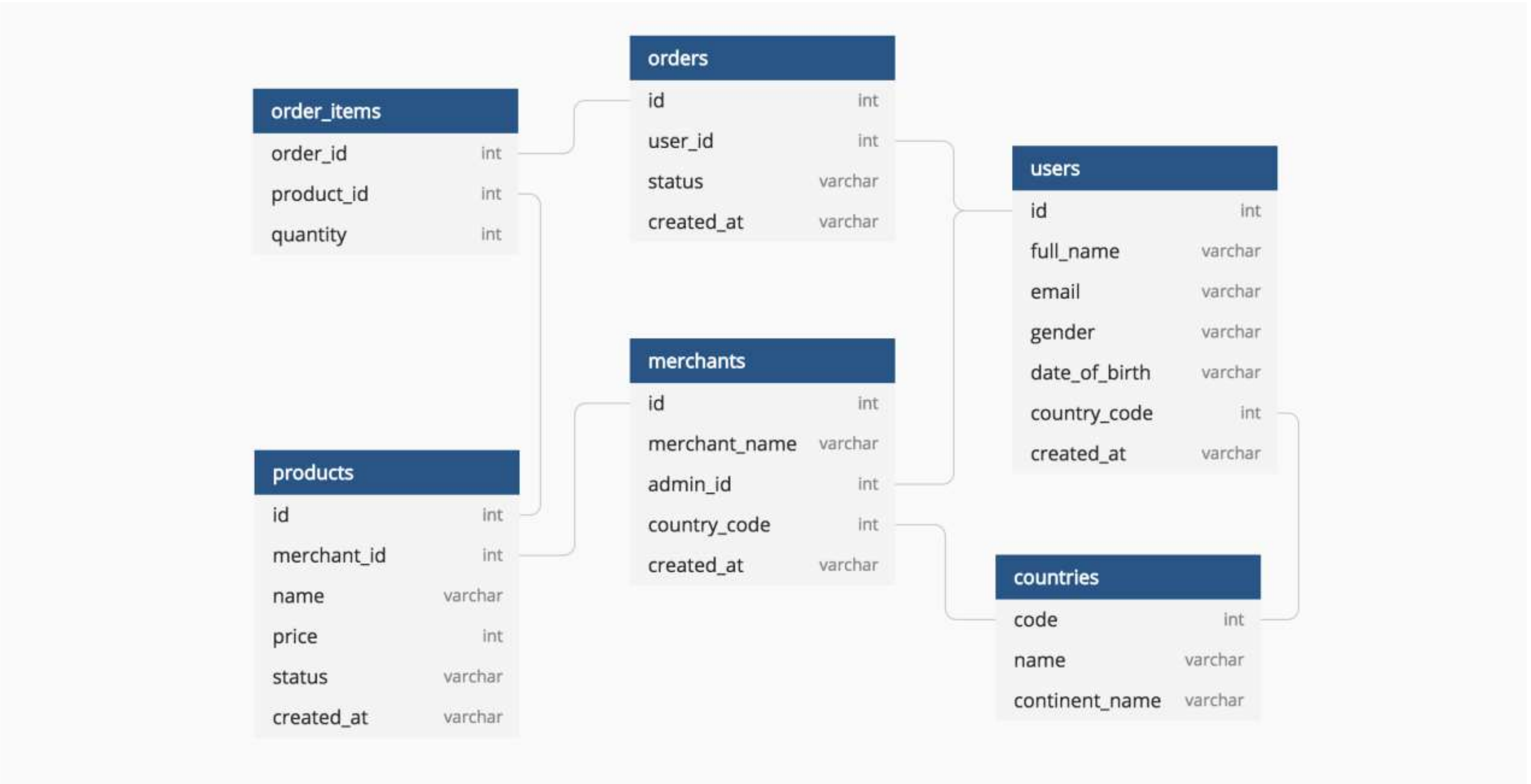
SECTION

Section_identifier	Course_number	Semester	Year	Instructor

GRADE_REPORT

Student_number	Section_identifier	Grade

Schema
databas



order_items	
order_id	int
product_id	int
quantity	int

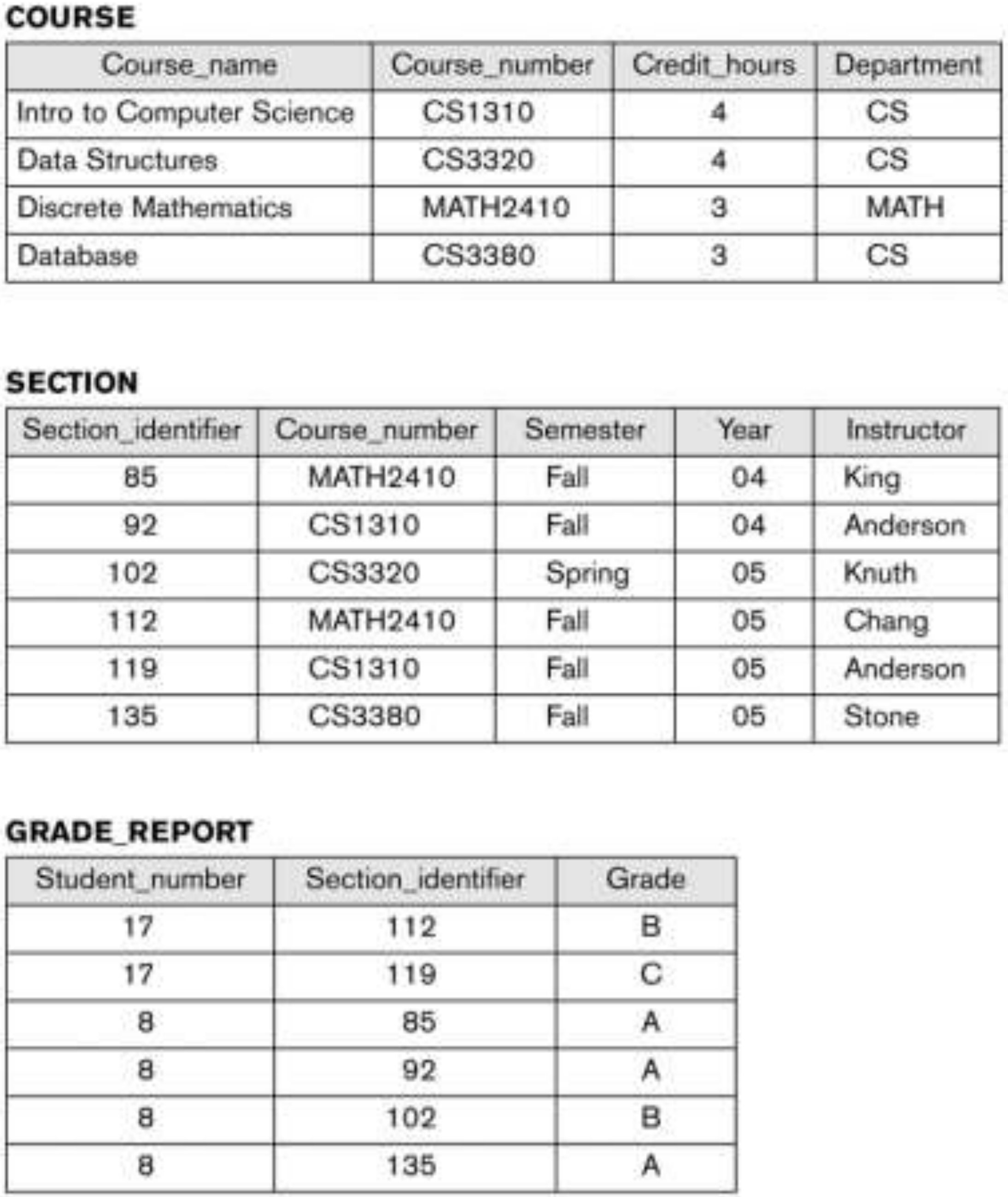
orders	
id	int
user_id	int
status	varchar
created_at	varchar

users	
id	int
full_name	varchar
email	varchar
gender	varchar
date_of_birth	varchar
country_code	int
created_at	varchar

merchants	
id	int
merchant_name	varchar
admin_id	int
country_code	int
created_at	varchar

products	
id	int
merchant_id	int
name	varchar
price	int
status	varchar
created_at	varchar

countries	
code	int
name	varchar
continent_name	varchar



COURSE

Course_name	Course_number	Credit_hours	Department
Intro to Computer Science	CS1310	4	CS
Data Structures	CS3320	4	CS
Discrete Mathematics	MATH2410	3	MATH
Database	CS3380	3	CS

SECTION

Section_identifier	Course_number	Semester	Year	Instructor
85	MATH2410	Fall	04	King
92	CS1310	Fall	04	Anderson
102	CS3320	Spring	05	Knuth
112	MATH2410	Fall	05	Chang
119	CS1310	Fall	05	Anderson
135	CS3380	Fall	05	Stone

GRADE_REPORT

Student_number	Section_identifier	Grade
17	112	B
17	119	C
8	85	A
8	92	A
8	102	B
8	135	A

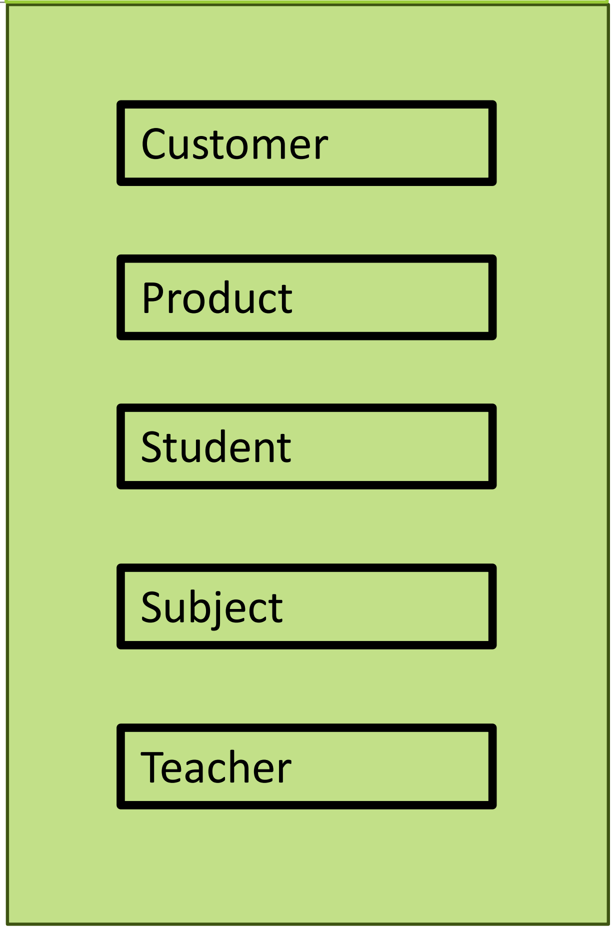
Entities

Entities: a distinct object, concept, or thing about which data can be stored. It represents a real-world item with characteristics that can be described and managed.

represented in a database by **tables**, where each **row** in the table represents a **specific instance** of the **entity**, and each **column** represents an **attribute** or property of that instance.

- • The basic building blocks of an ER diagram
- • Represent various real world notions, such as people, places, objects, events, items, and other concepts
- • Within one ERD each entity must have a different name

Example:



ATTRIBUTES

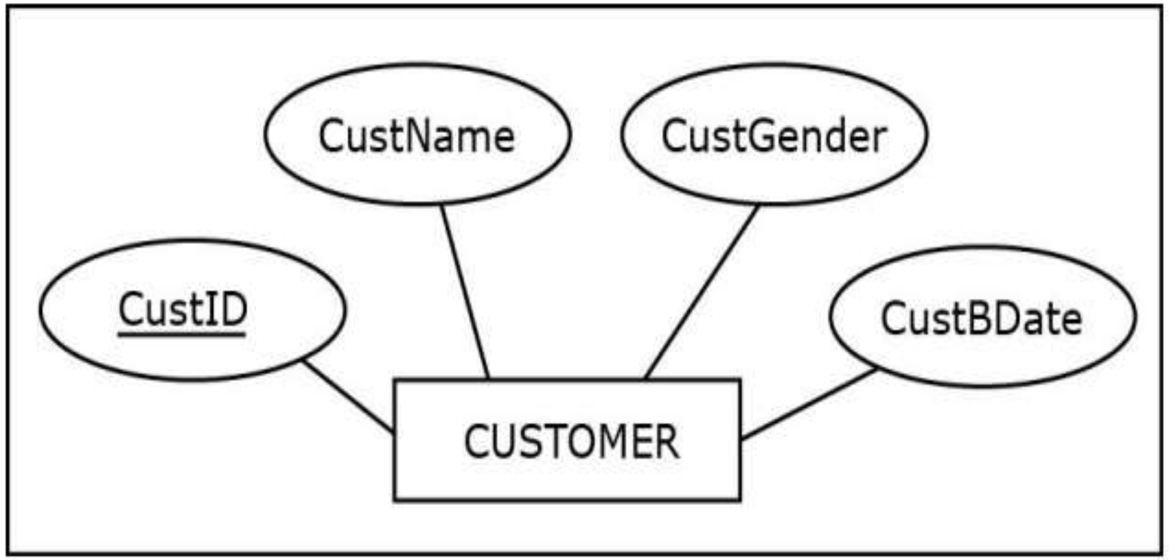
Attribute - **depiction** of a characteristic of an **entity**

• Represents the **details** that will be recorded for each entity instance

• *Within one entity, each attribute must have a different name*

Unique Attribute - attribute whose value is different for each entity instance

• *Every entity must have at least one unique attribute.*

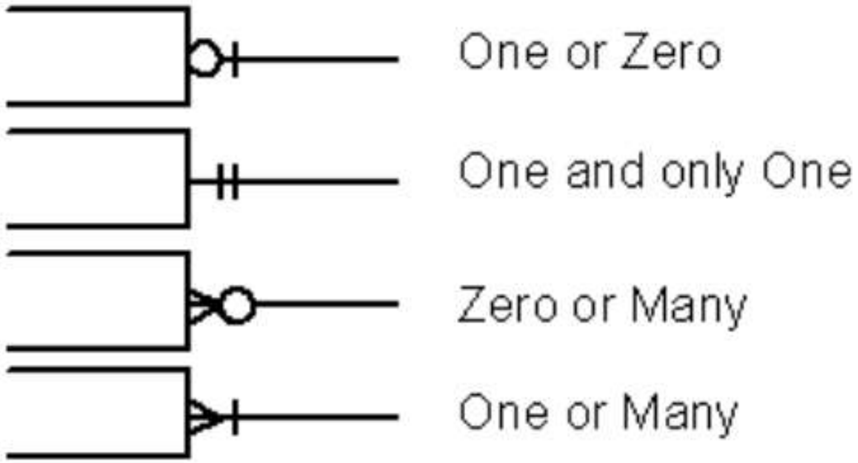


RELATIONSHIPS

- **Relationship** - ER modeling construct depicting how entities are related
- Within an ER diagram, each **entity** must be related to at least one other.
- Cardinality constraints - depict how many

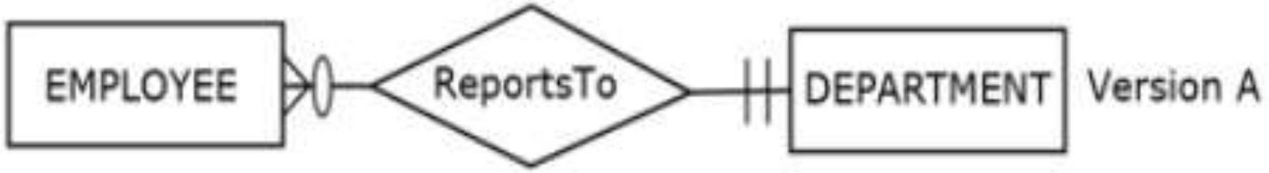
instances of one entity can be associated with instances of another entity

- • **Max**imum cardinality
  - o **One** (represented by a straight bar: **I**)
  - o **Many** (represented by a crow's foot symbol: >)
- • **Min**imum cardinality (participation)
  - o **Optional** (represented by a circular symbol: **0**)
  - o **Mandatory** (represented by a straight bar: **I**)



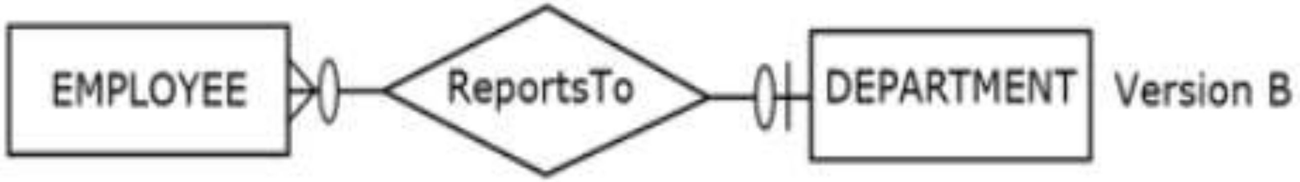
Version A:

- *Each employee reports to exactly one department. Each department has between zero and many employees reporting to it.*



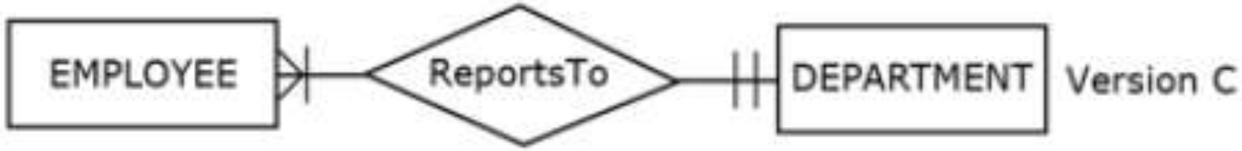
Version B:

- *An employee can report to one department or to no departments at all. Each department has between zero and many employees reporting to it.*



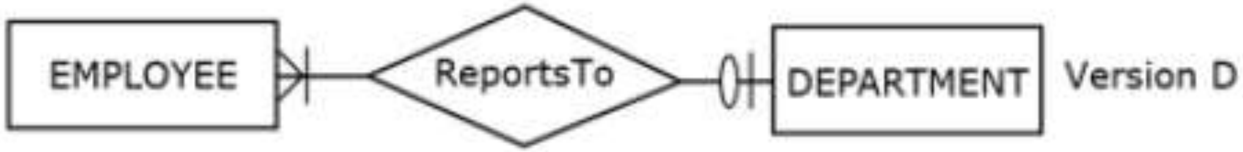
Version C:

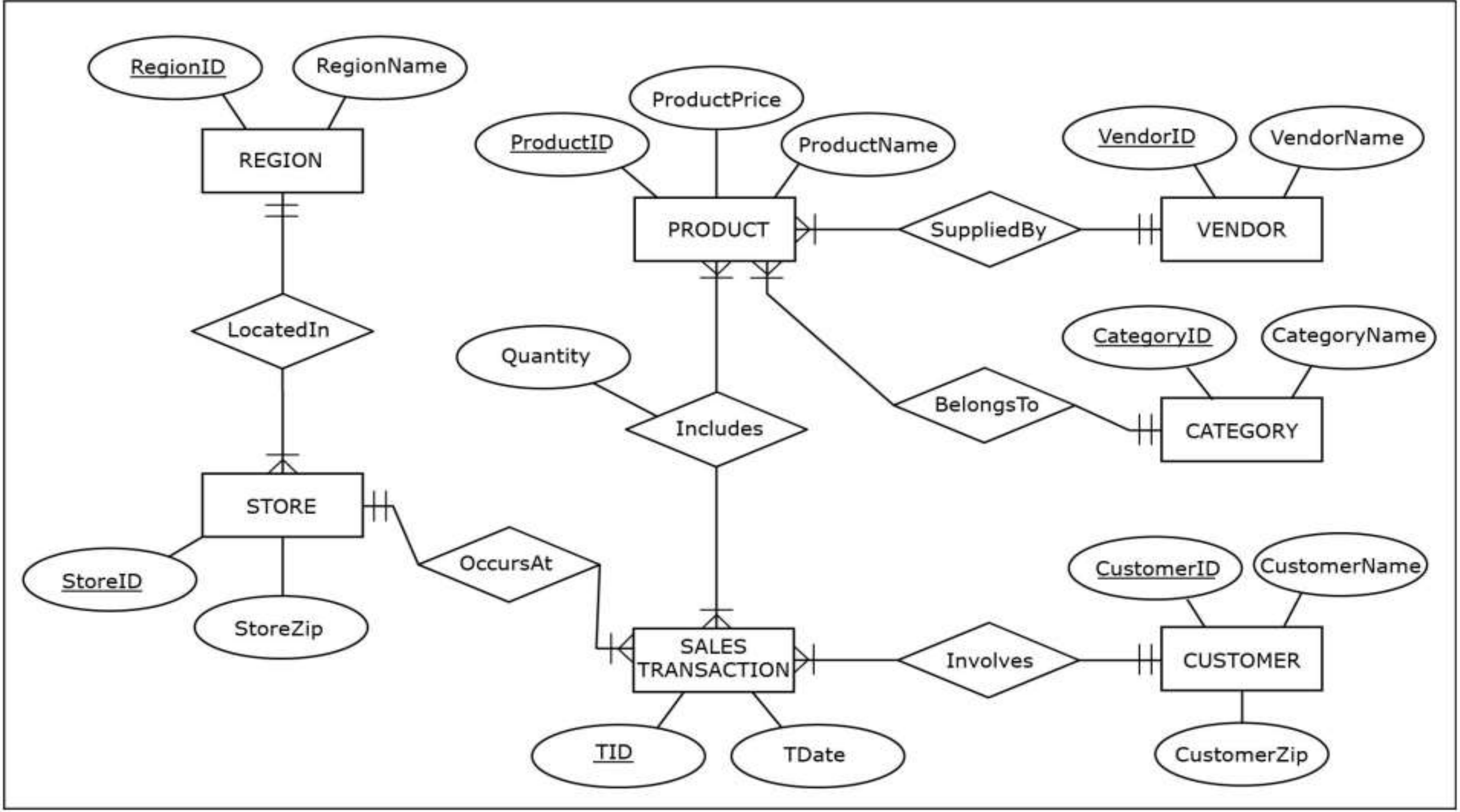
- *Each employee reports to exactly one department. A department must have at least one employee reporting to it, but it may have many employees reporting to it.*

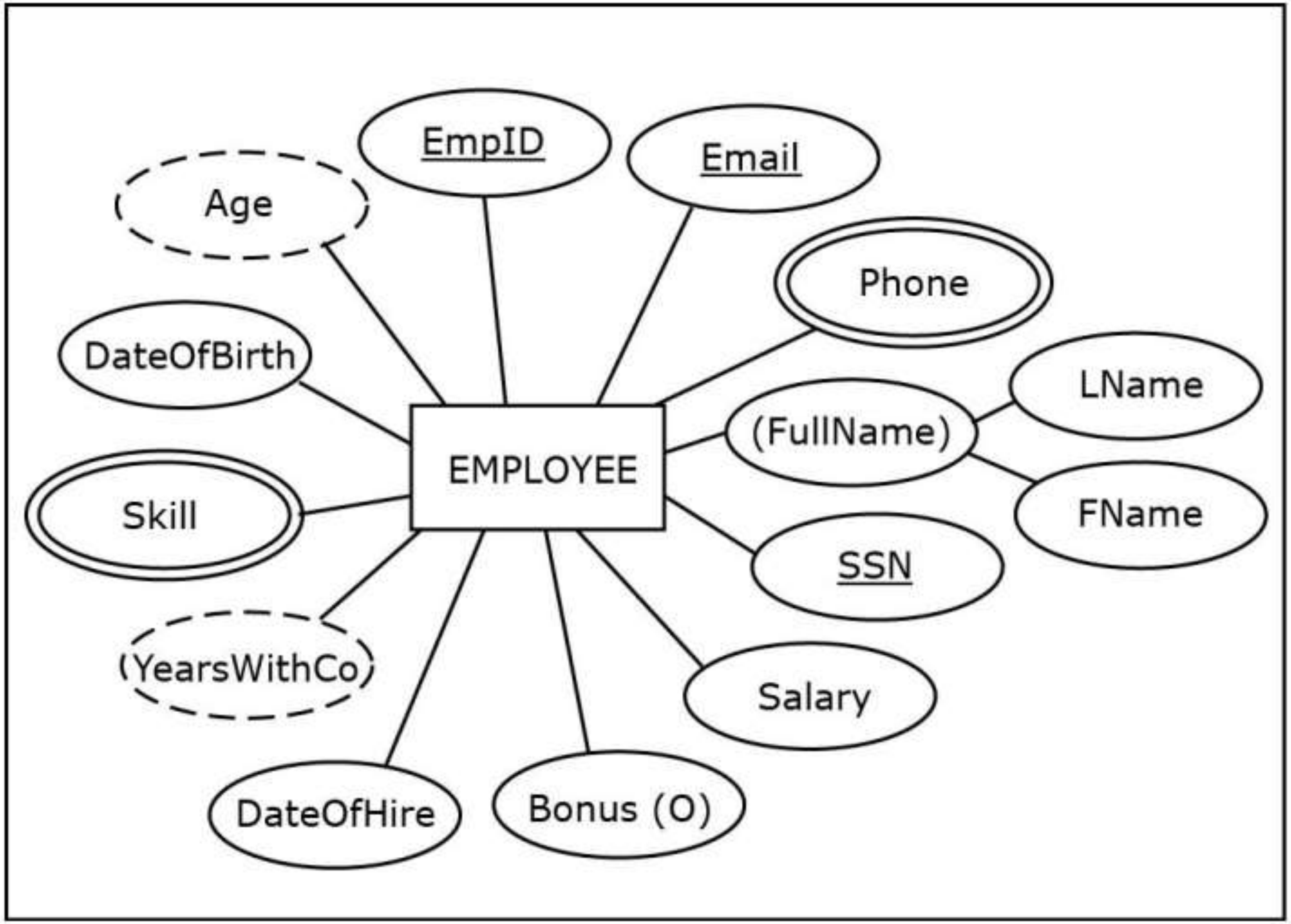


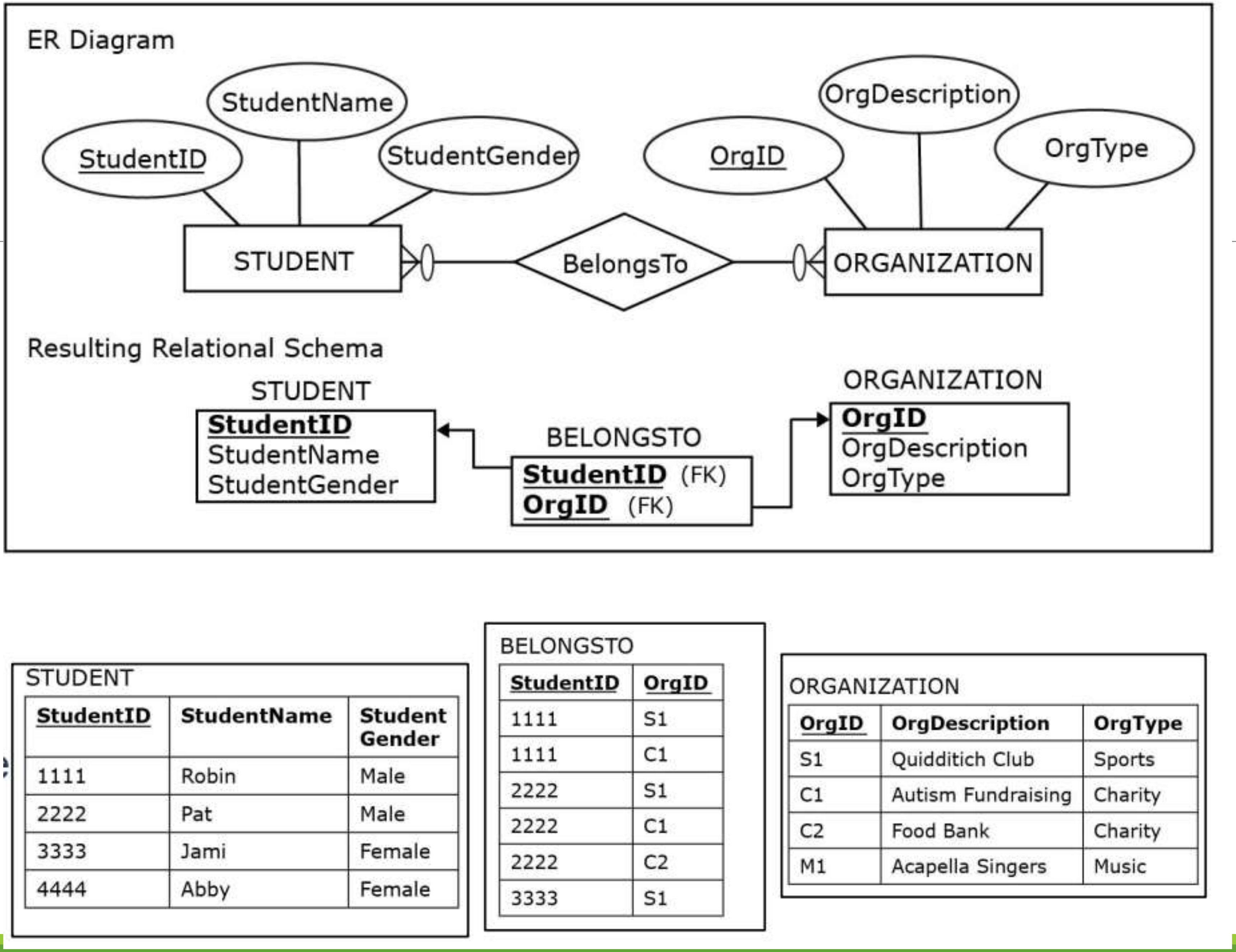
Version D:

- *An employee can report to one department or to no departments at all. A department must have at least one employee reporting to it, but it may have many employees reporting to it.*




RegionID
RegionName
REGION
LocatedIn
STORE
StoreID
StoreZip
ProductPrice
ProductID
ProductName
PRODUCT
SuppliedBy
VendorID
VendorName
VENDOR
CategoryID
CategoryName
BelongsTo
CATEGORY
Quantity
Includes
OccursAt
SALES TRANSACTION
TID
TDate
Involves
CustomerID
CustomerName
CUSTOMER
CustomerZip


EmpID
Email
Age
Phone
DateOfBirth
LName
(FullName)
EMPLOYEE
FName
Skill
SSN
(YearsWithCo)
Salary
DateOfHire
Bonus (O)



STUDENT

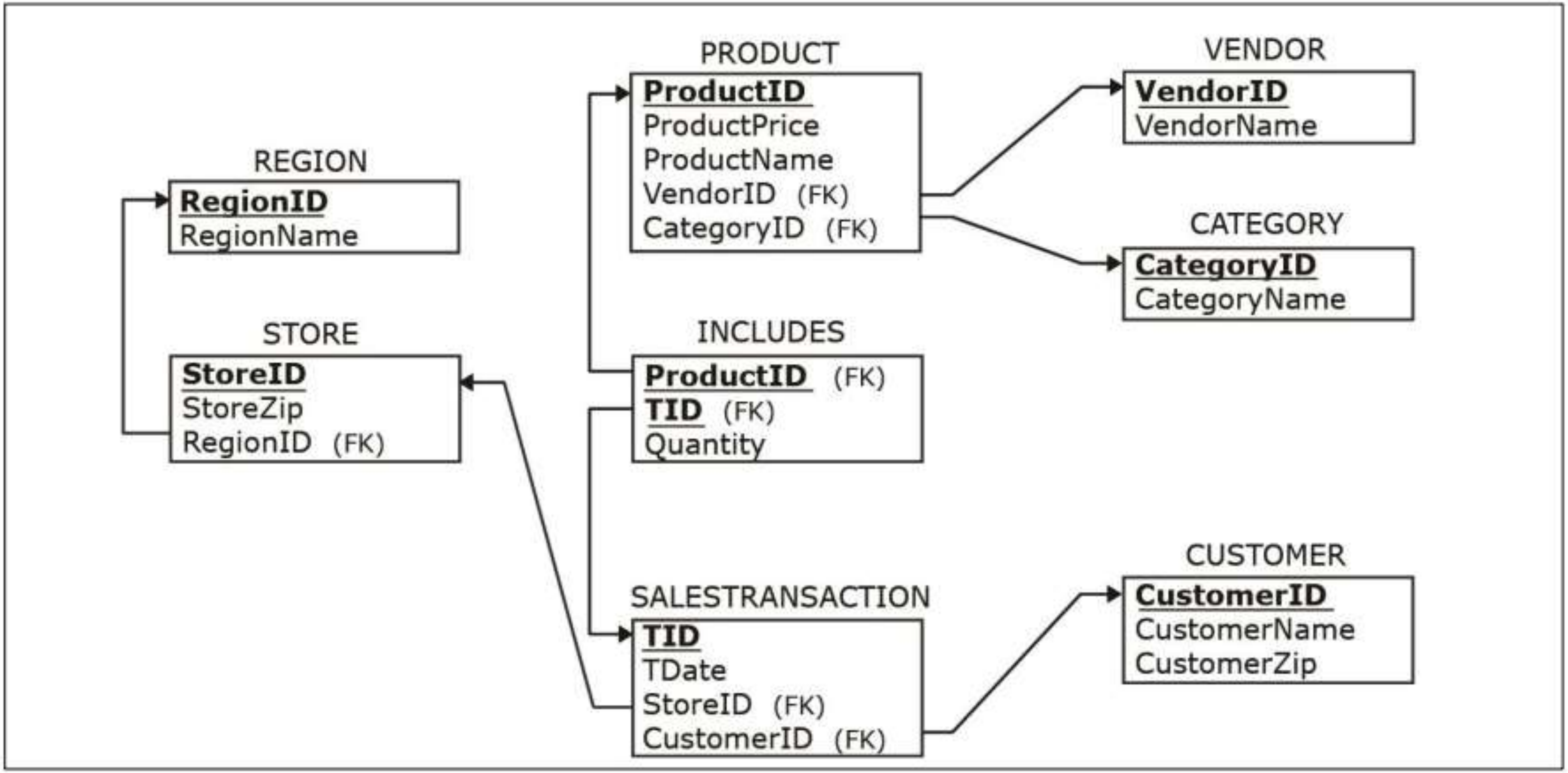
StudentID	**StudentName**	**Student Gender**
1111	Robin	Male
2222	Pat	Male
3333	Jami	Female
4444	Abby	Female

BELONGSTO

StudentID	**OrgID**
1111	S1
1111	C1
2222	S1
2222	C1
2222	C2
3333	S1

ORGANIZATION

OrgID	**OrgDescription**	**OrgType**
S1	Quidditich Club	Sports
C1	Autism Fundraising	Charity
C2	Food Bank	Charity
M1	Acapella Singers	Music



REGION

<u>RegionID</u>
RegionName

STORE

<u>StoreID</u>
StoreZip
RegionID (FK)

PRODUCT

<u>ProductID</u>
ProductPrice
ProductName
VendorID (FK)
CategoryID (FK)

INCLUDES

<u>ProductID</u> (FK)
<u>TID</u> (FK)
Quantity

SALESTRANSACTION

<u>TID</u>
TDate
StoreID (FK)
CustomerID (FK)

VENDOR

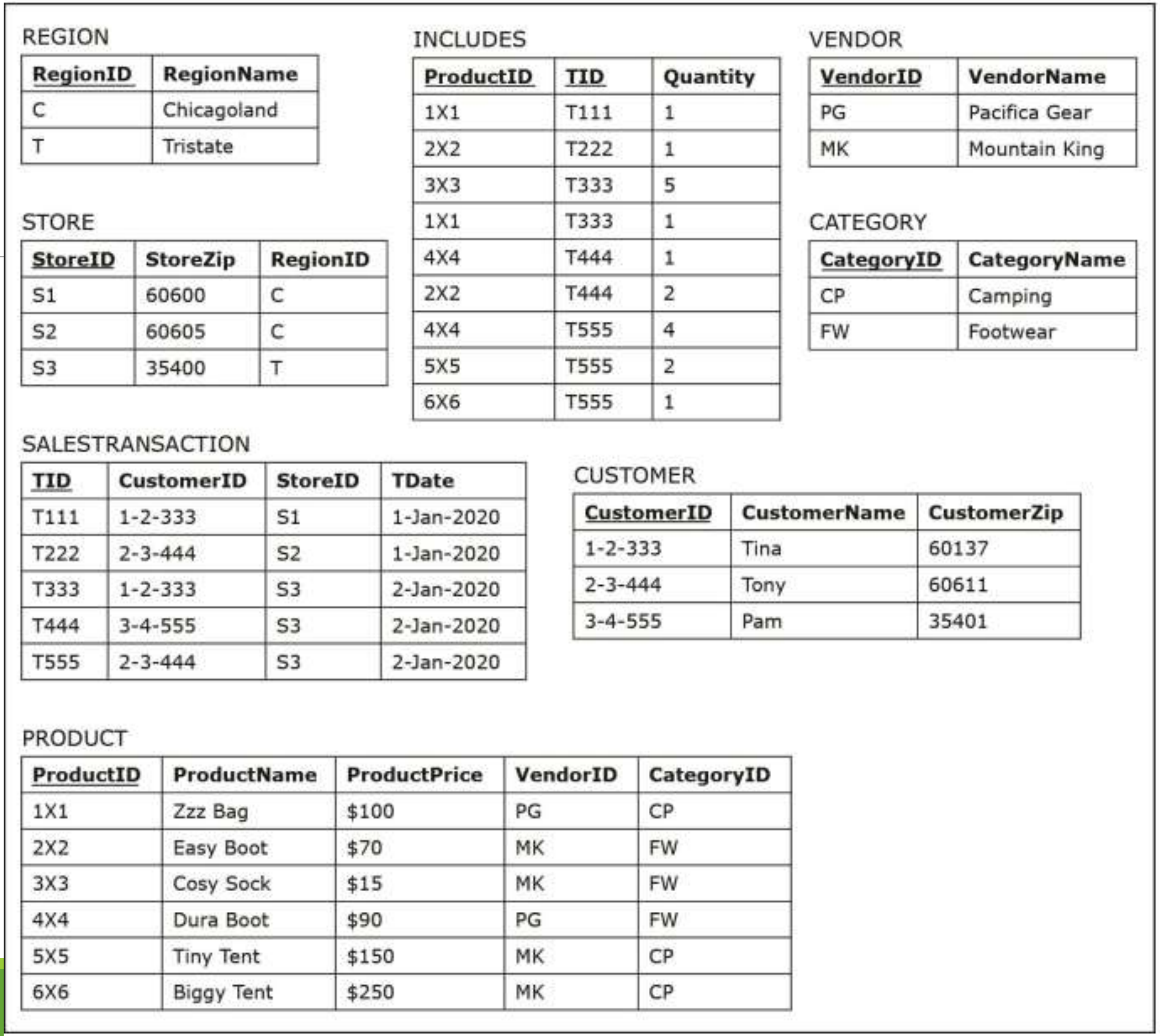
<u>VendorID</u>
VendorName

CATEGORY

<u>CategoryID</u>
CategoryName

CUSTOMER

<u>CustomerID</u>
CustomerName
CustomerZip



REGION

RegionID	RegionName
C	Chicagoland
T	Tristate

INCLUDES

ProductID	TID	Quantity
1X1	T111	1
2X2	T222	1
3X3	T333	5
1X1	T333	1
4X4	T444	1
2X2	T444	2
4X4	T555	4
5X5	T555	2
6X6	T555	1

VENDOR

VendorID	VendorName
PG	Pacifica Gear
MK	Mountain King

STORE

StoreID	StoreZip	RegionID
S1	60600	C
S2	60605	C
S3	35400	T

CATEGORY

CategoryID	CategoryName
CP	Camping
FW	Footwear

SALESTRANSACTION

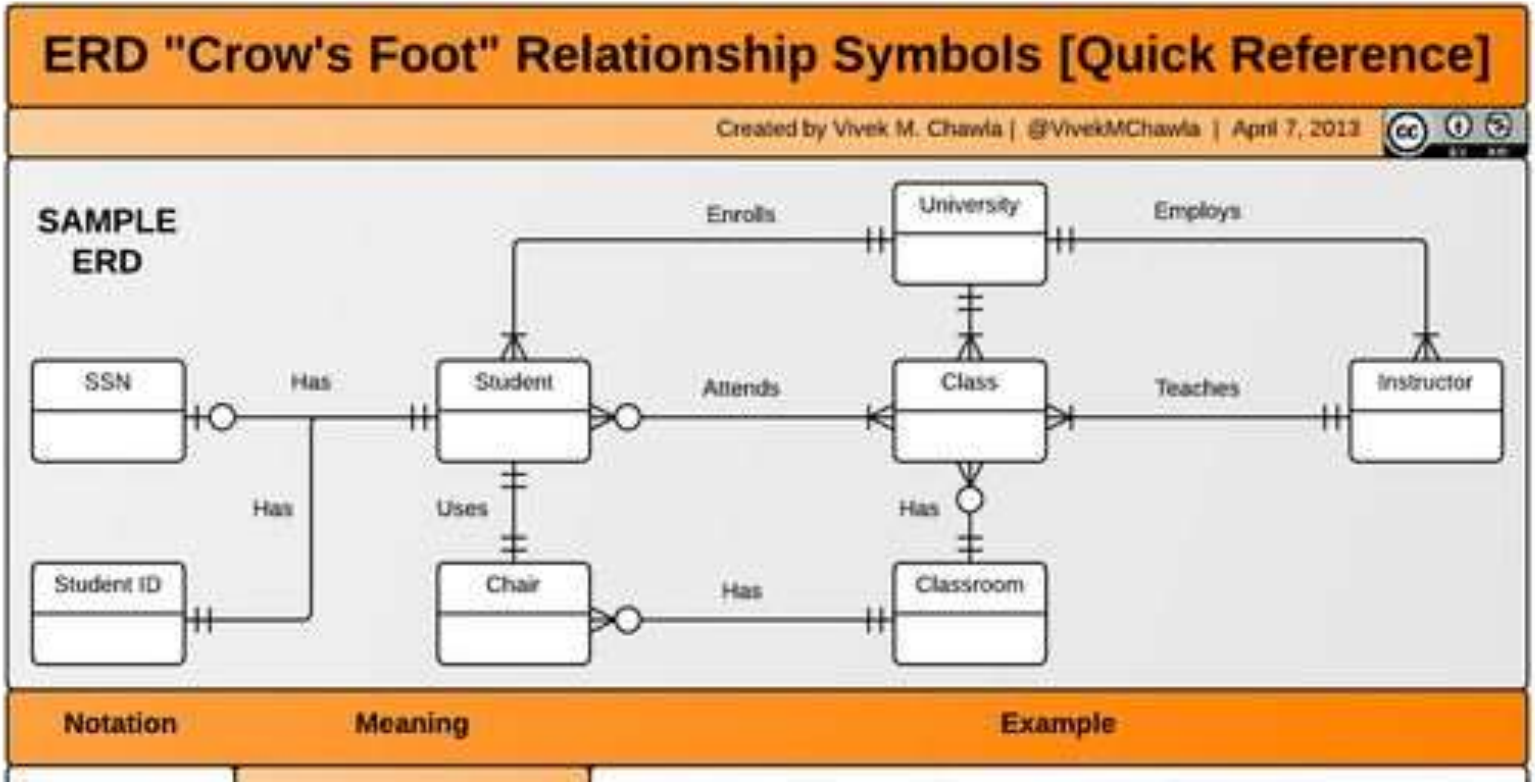
TID	CustomerID	StoreID	TDate
T111	1-2-333	S1	1-Jan-2020
T222	2-3-444	S2	1-Jan-2020
T333	1-2-333	S3	2-Jan-2020
T444	3-4-555	S3	2-Jan-2020
T555	2-3-444	S3	2-Jan-2020

CUSTOMER

CustomerID	CustomerName	CustomerZip
1-2-333	Tina	60137
2-3-444	Tony	60611
3-4-555	Pam	35401

PRODUCT

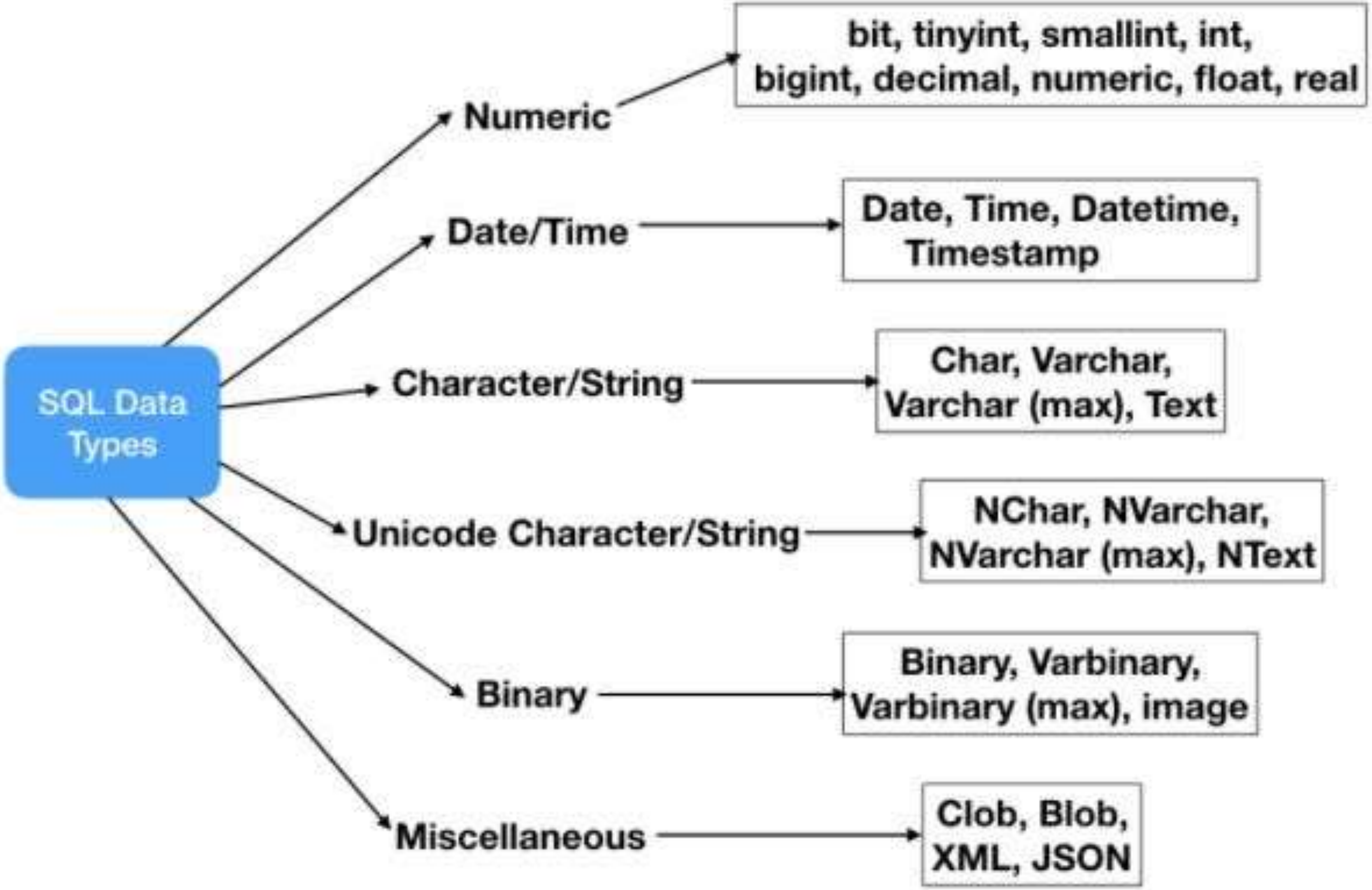
ProductID	ProductName	ProductPrice	VendorID	CategoryID
1X1	Zzz Bag	$100	PG	CP
2X2	Easy Boot	$70	MK	FW
3X3	Cosy Sock	$15	MK	FW
4X4	Dura Boot	$90	PG	FW
5X5	Tiny Tent	$150	MK	CP
6X6	Biggy Tent	$250	MK	CP



ERD "Crow's Foot" Relationship Symbols [Quick Reference]

Created by Vivek M. Chawla | @VivekMChawla | April 7, 2013

Notation	Meaning	Example



SQL Data Types

Numeric
bit, tinyint, smallint, int, bigint, decimal, numeric, float, real

Date/Time
Date, Time, Datetime, Timestamp

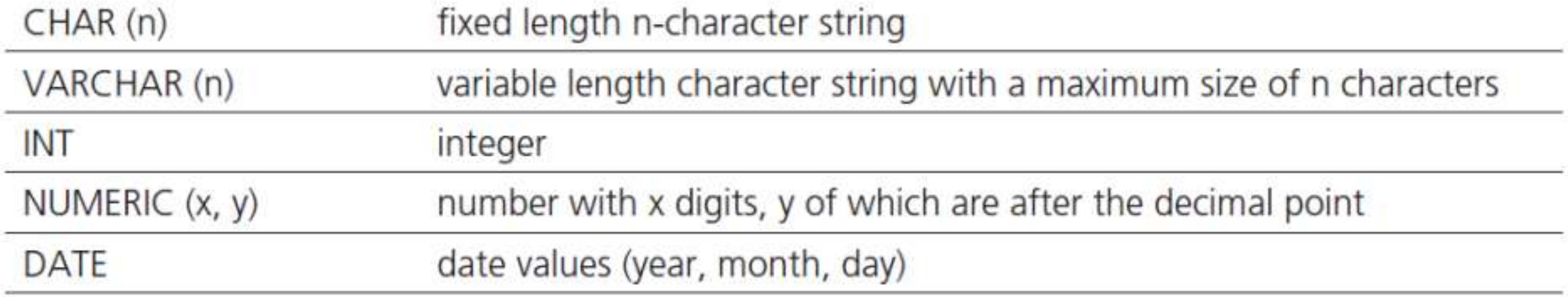
Character/String
Char, Varchar, Varchar (max), Text

Unicode Character/String
NChar, NVarchar, NVarchar (max), NText

Binary
Binary, Varbinary, Varbinary (max), image

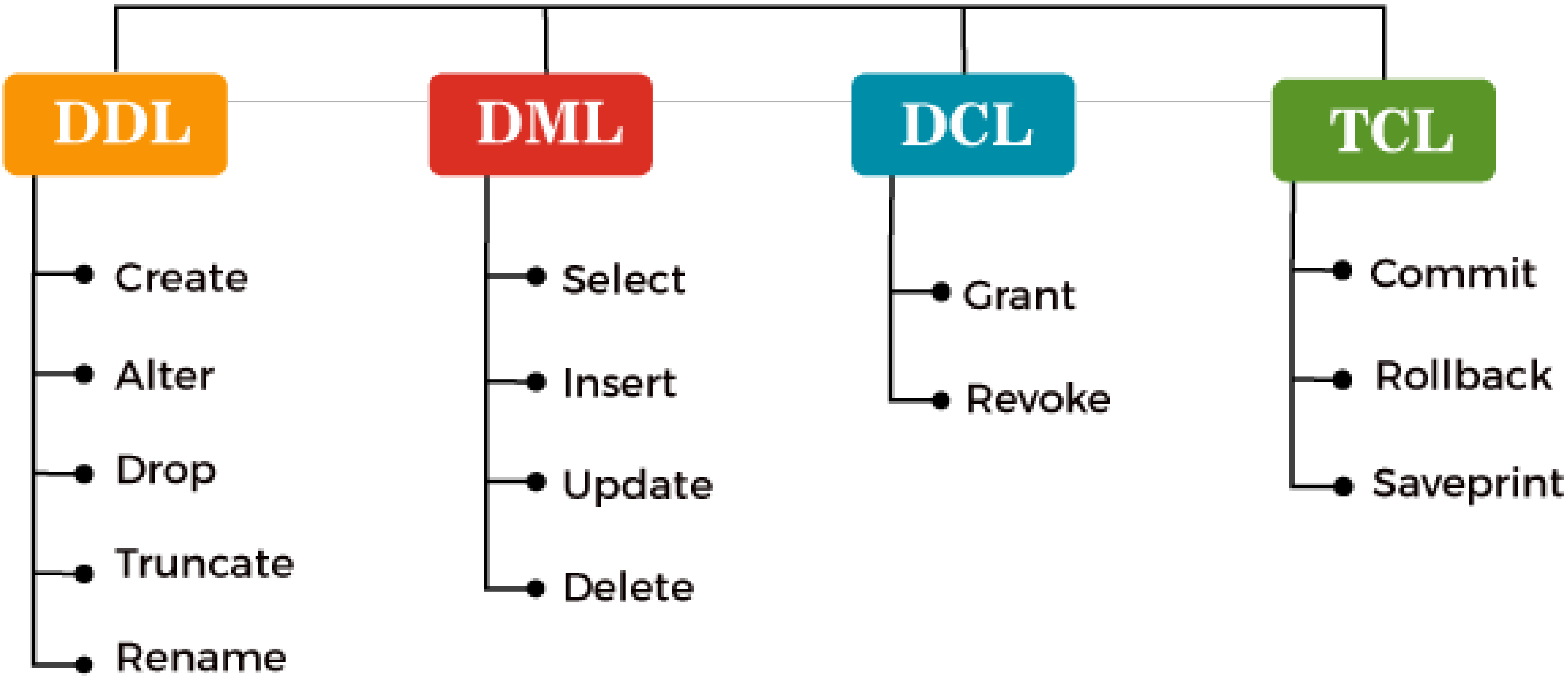
Miscellaneous
Clob, Blob, XML, JSON

- **SQL data types**
  - Each column of each SQL created relation has a specified data type
  - Commonly used SQL data types:

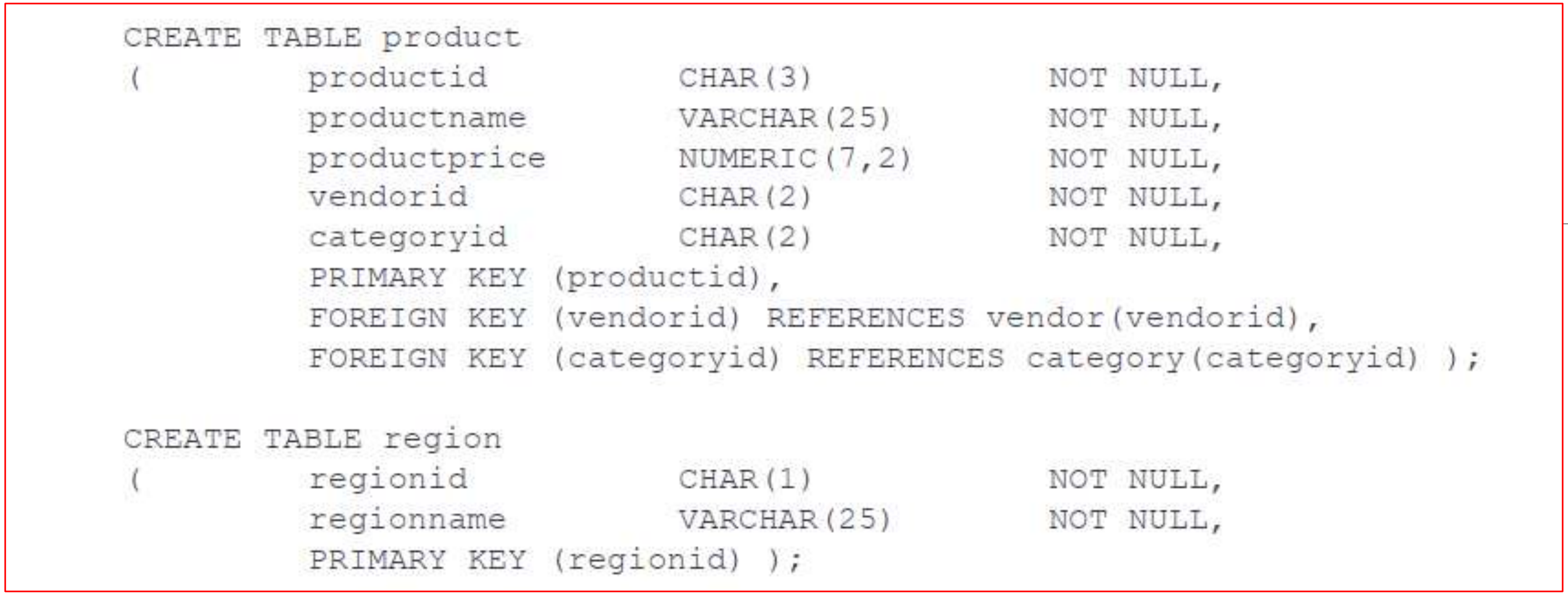


CHAR (n)	fixed length n-character string
VARCHAR (n)	variable length character string with a maximum size of n characters
INT	integer
NUMERIC (x, y)	number with x digits, y of which are after the decimal point
DATE	date values (year, month, day)

Types of SQL Commands

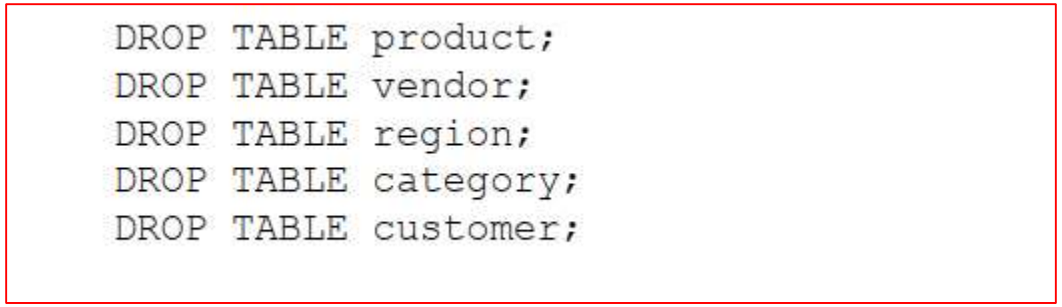


- DDL
  - Create
  - Alter
  - Drop
  - Truncate
  - Rename
- DML
  - Select
  - Insert
  - Update
  - Delete
- DCL
  - Grant
  - Revoke
- TCL
  - Commit
  - Rollback
  - Saveprint



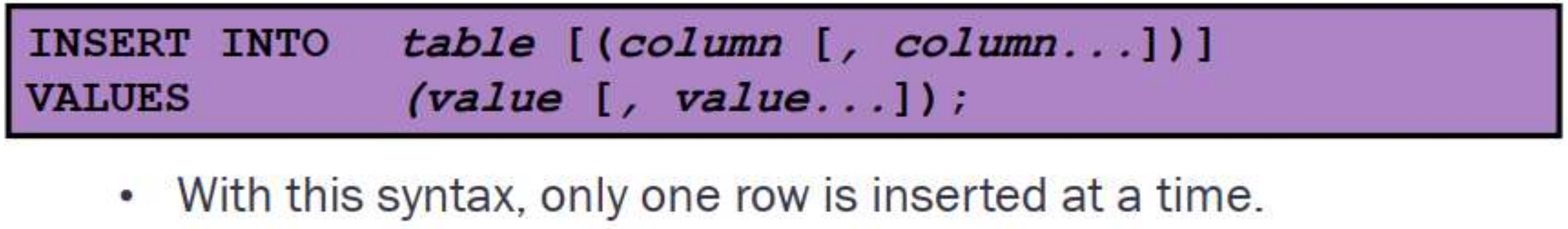
```
CREATE TABLE product
(         productid          CHAR(3)            NOT NULL,
          productname        VARCHAR(25)        NOT NULL,
          productprice       NUMERIC(7,2)       NOT NULL,
          vendorid           CHAR(2)            NOT NULL,
          categoryid         CHAR(2)            NOT NULL,
          PRIMARY KEY (productid),
          FOREIGN KEY (vendorid) REFERENCES vendor(vendorid),
          FOREIGN KEY (categoryid) REFERENCES category(categoryid) );

CREATE TABLE region
(         regionid           CHAR(1)            NOT NULL,
          regionname         VARCHAR(25)        NOT NULL,
          PRIMARY KEY (regionid) );
```



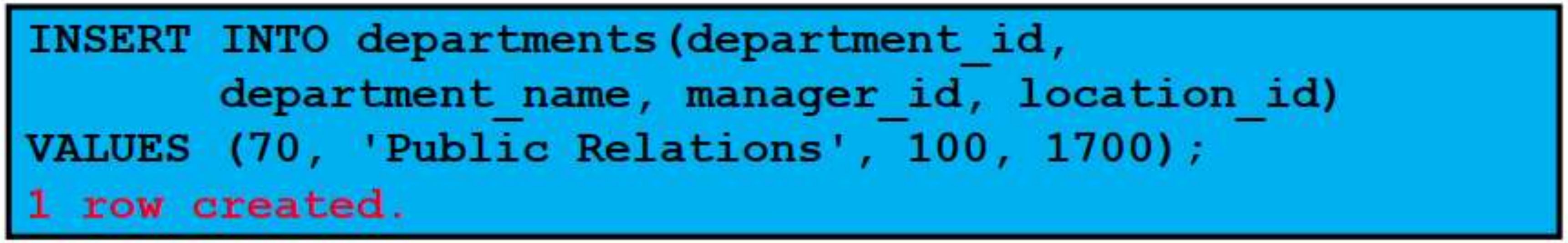
```
DROP TABLE product;
DROP TABLE vendor;
DROP TABLE region;
DROP TABLE category;
DROP TABLE customer;
```

Insert into - values

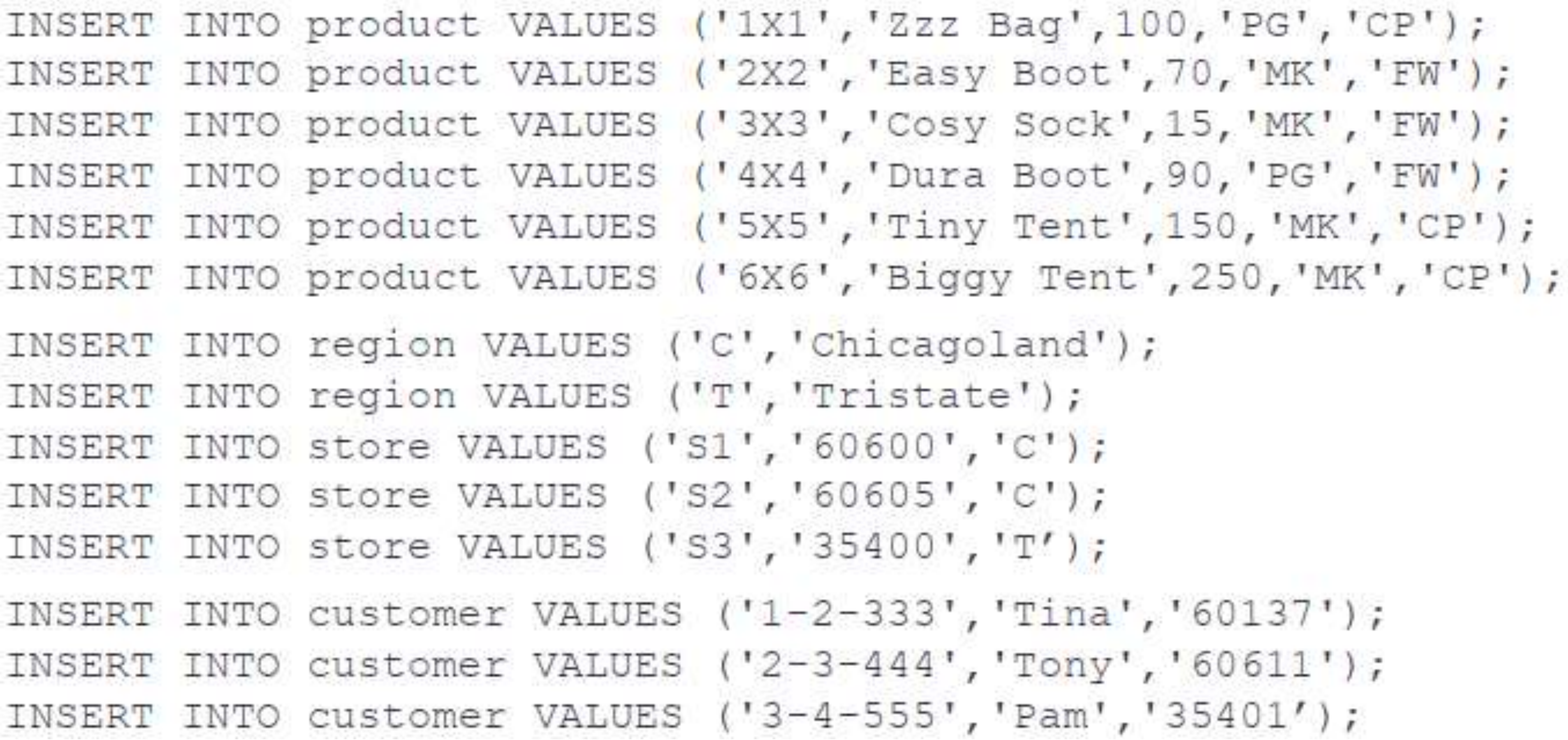


```
INSERT INTO   table [(column [, column...])]
VALUES        (value [, value...]);
```

- With this syntax, only one row is inserted at a time.

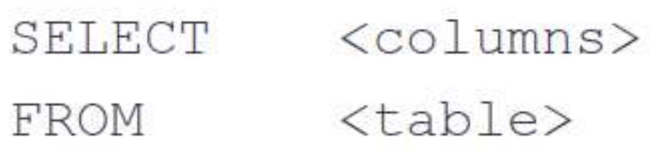


```
INSERT INTO departments(department_id,
       department_name, manager_id, location_id)
VALUES (70, 'Public Relations', 100, 1700);
1 row created.
```



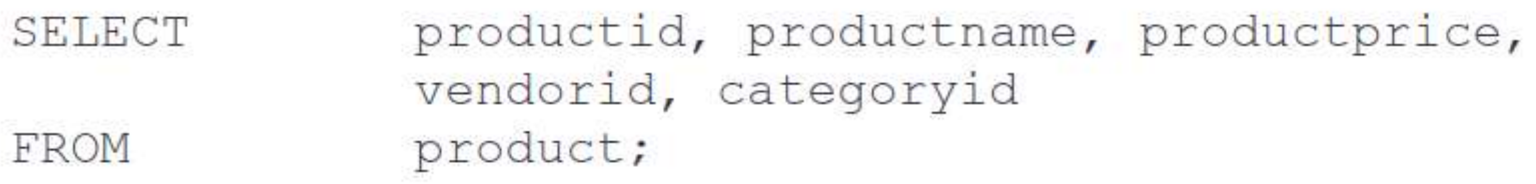
```
INSERT INTO product VALUES ('1X1','Zzz Bag',100,'PG','CP');
INSERT INTO product VALUES ('2X2','Easy Boot',70,'MK','FW');
INSERT INTO product VALUES ('3X3','Cosy Sock',15,'MK','FW');
INSERT INTO product VALUES ('4X4','Dura Boot',90,'PG','FW');
INSERT INTO product VALUES ('5X5','Tiny Tent',150,'MK','CP');
INSERT INTO product VALUES ('6X6','Biggy Tent',250,'MK','CP');
INSERT INTO region VALUES ('C','Chicagoland');
INSERT INTO region VALUES ('T','Tristate');
INSERT INTO store VALUES ('S1','60600','C');
INSERT INTO store VALUES ('S2','60605','C');
INSERT INTO store VALUES ('S3','35400','T’);
INSERT INTO customer VALUES ('1-2-333','Tina','60137');
INSERT INTO customer VALUES ('2-3-444','Tony','60611');
INSERT INTO customer VALUES ('3-4-555','Pam','35401’);
```

Select - From

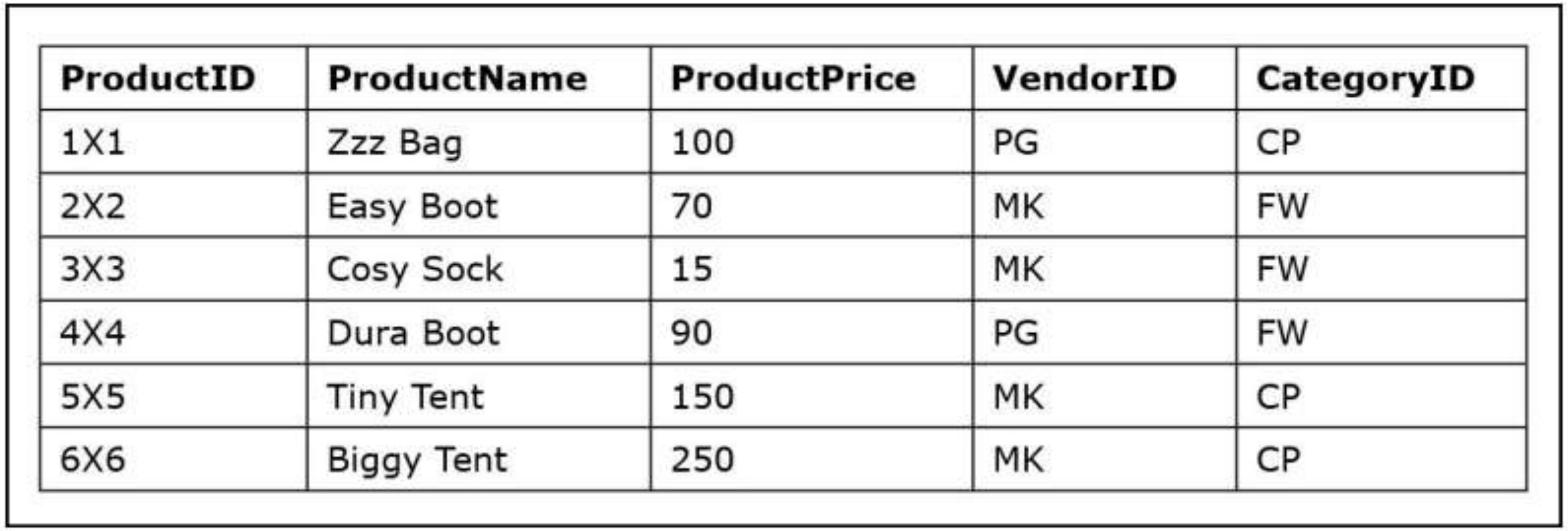


```
SELECT     <columns>
FROM       <table>
```

```
SELECT          *
FROM            product;
```

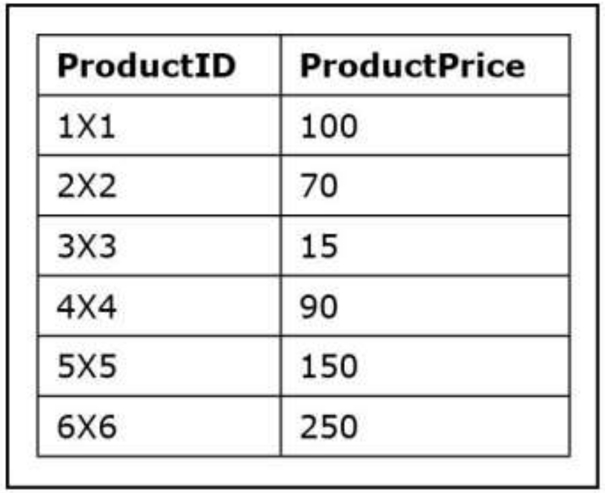


```
SELECT        productid, productname, productprice,
              vendorid, categoryid
FROM          product;
```

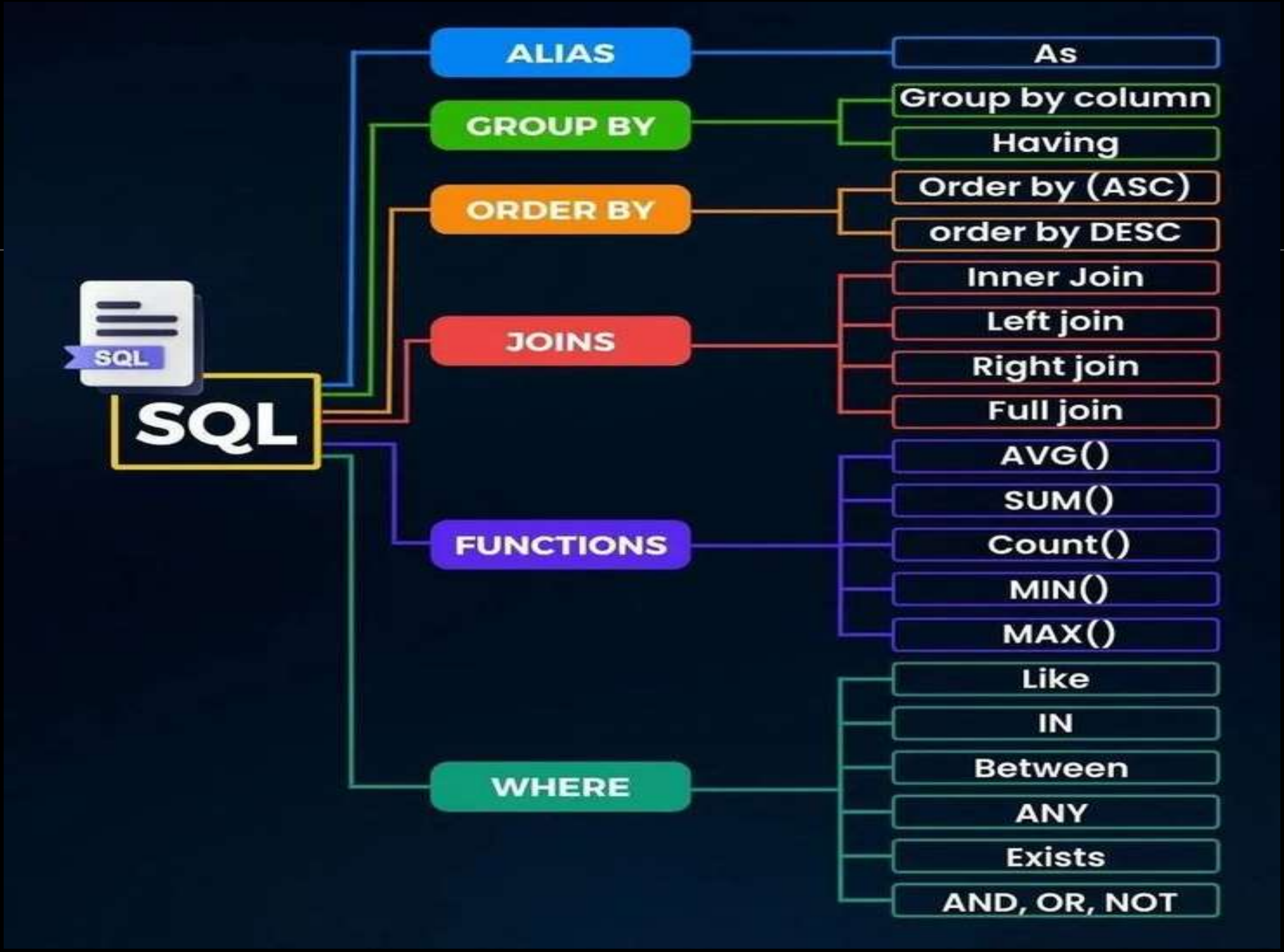


ProductID	ProductName	ProductPrice	VendorID	CategoryID
1X1	Zzz Bag	100	PG	CP
2X2	Easy Boot	70	MK	FW
3X3	Cosy Sock	15	MK	FW
4X4	Dura Boot	90	PG	FW
5X5	Tiny Tent	150	MK	CP
6X6	Biggy Tent	250	MK	CP

```
SELECT         productid, productprice
FROM           product;
```



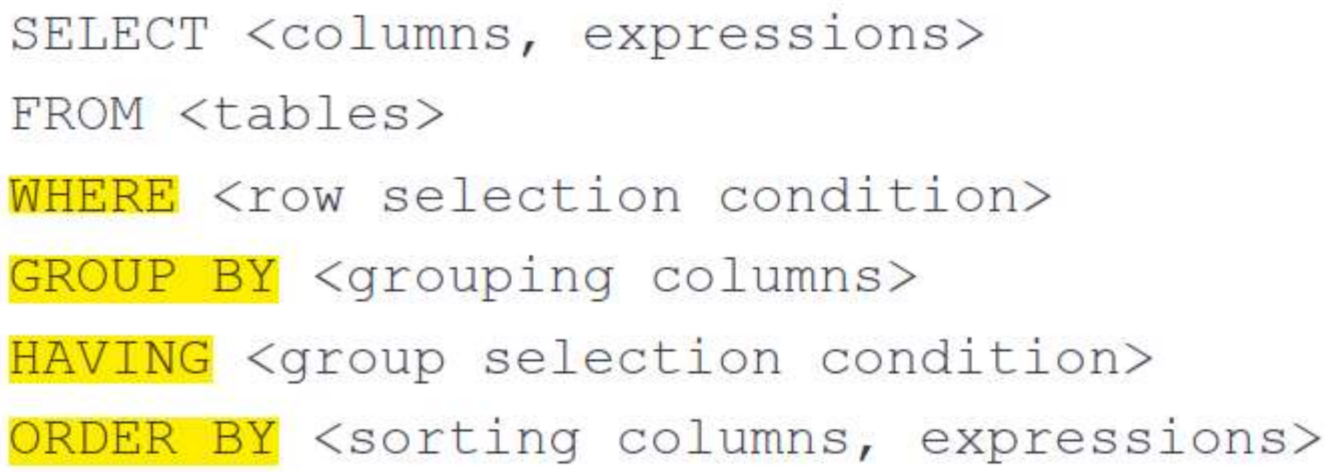
ProductID	ProductPrice
1X1	100
2X2	70
3X3	15
4X4	90
5X5	150
6X6	250



SQL

- ALIAS
  - As
- GROUP BY
  - Group by column
  - Having
- ORDER BY
  - Order by (ASC)
  - order by DESC
- JOINS
  - Inner Join
  - Left join
  - Right join
  - Full join
- FUNCTIONS
  - AVG()
  - SUM()
  - Count()
  - MIN()
  - MAX()
- WHERE
  - Like
  - IN
  - Between
  - ANY
  - Exists
  - AND, OR, NOT

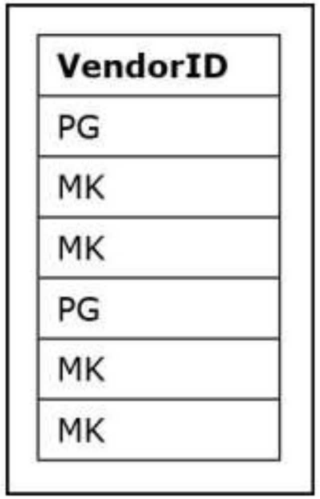
- **SELECT**
  - The SELECT FROM statement can contain other optional keywords, such as WHERE, GROUP BY, HAVING, and ORDER BY, appearing in this order: :



```
SELECT <columns, expressions>
FROM <tables>
WHERE <row selection condition>
GROUP BY <grouping columns>
HAVING <group selection condition>
ORDER BY <sorting columns, expressions>
```

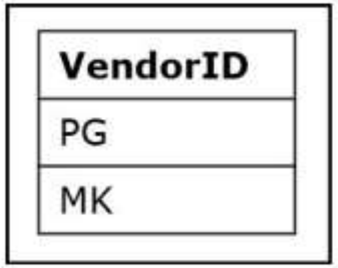
DISTINCT

```
SELECT      vendorid
FROM        product;
```



VendorID
PG
MK
MK
PG
MK
MK

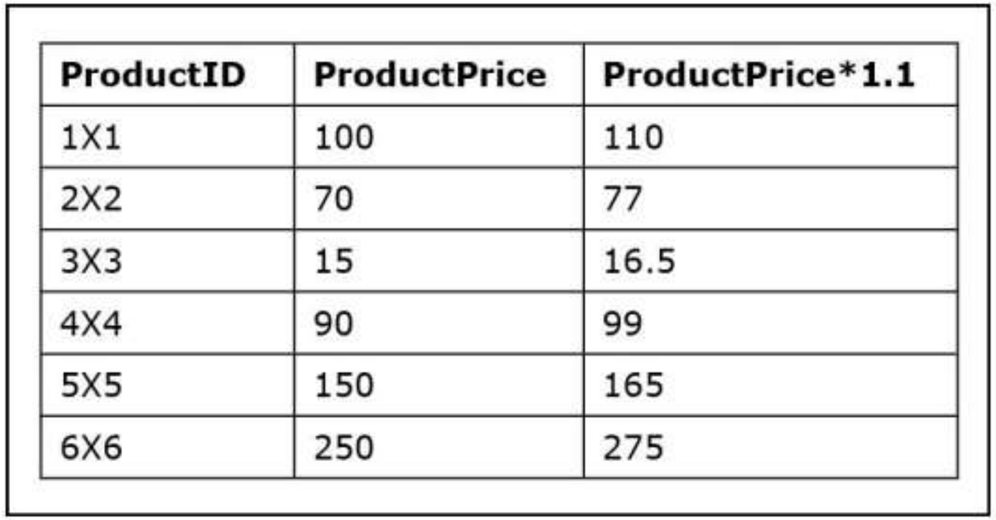
```
SELECT      DISTINCT vendorid
FROM        product;
```



VendorID
PG
MK

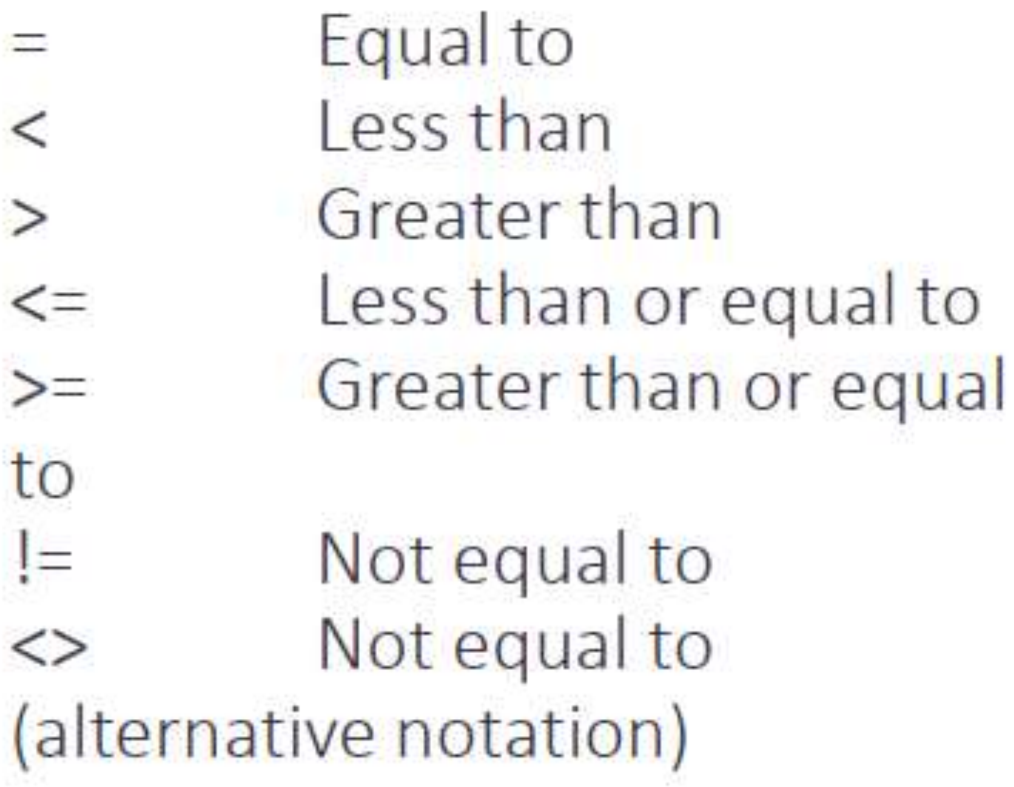
```
SELECT   productid, productprice, productprice * 1.1
FROM     product;
```

result:



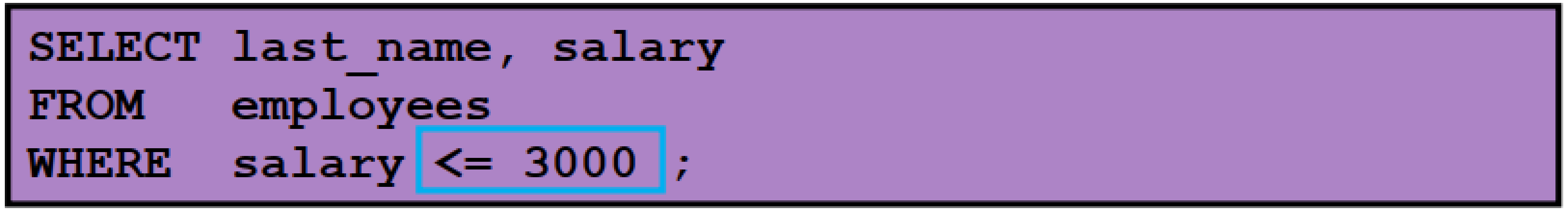
ProductID	ProductPrice	ProductPrice*1.1
1X1	100	110
2X2	70	77
3X3	15	16.5
4X4	90	99
5X5	150	165
6X6	250	275

Where

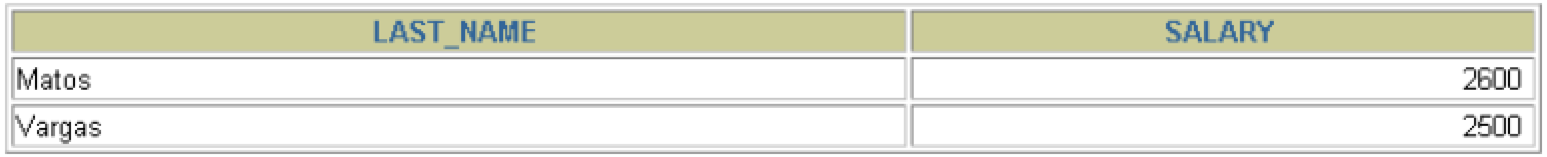


=	Equal to
<	Less than
>	Greater than
<=	Less than or equal to
>=	Greater than or equal to
!=	Not equal to
<>	Not equal to (alternative notation)

Ex1:

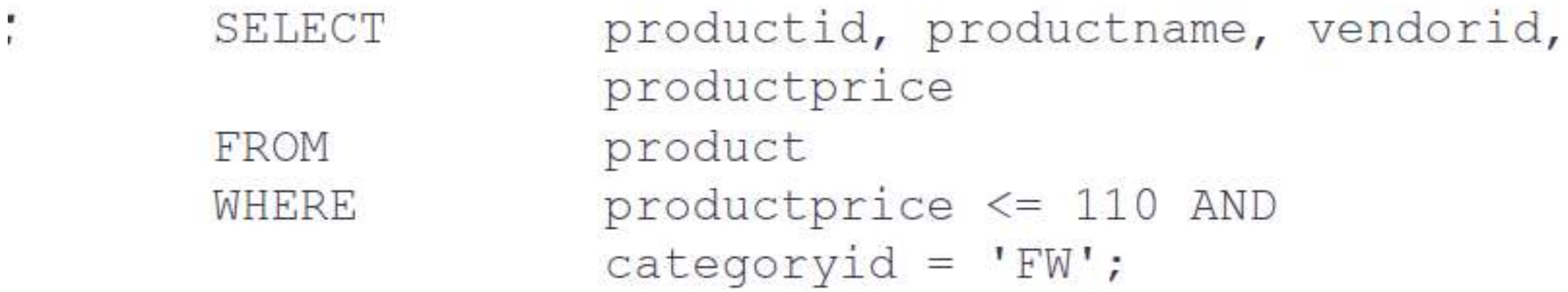


```
SELECT last_name, salary
FROM   employees
WHERE  salary <= 3000 ;
```



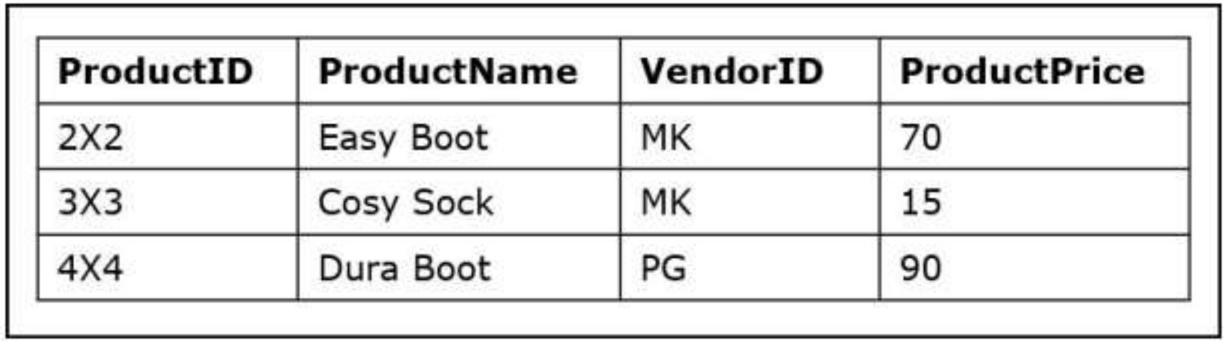
LAST_NAME	SALARY
Matos	2600
Vargas	2500

Ex2:



```
:       SELECT      productid, productname, vendorid,
                    productprice
        FROM        product
        WHERE       productprice <= 110 AND
                    categoryid = 'FW';
```

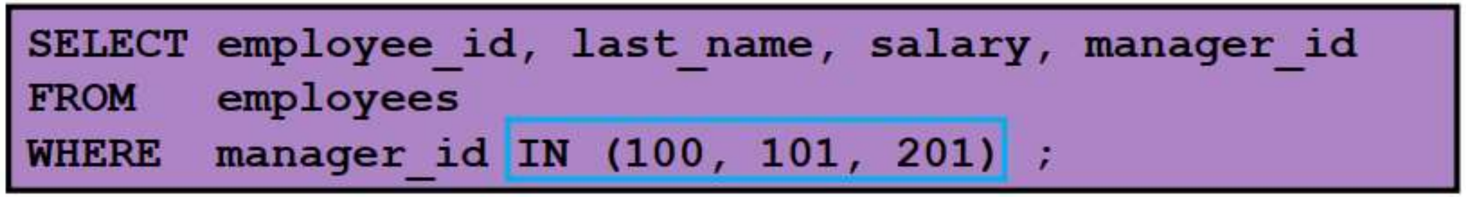
result:



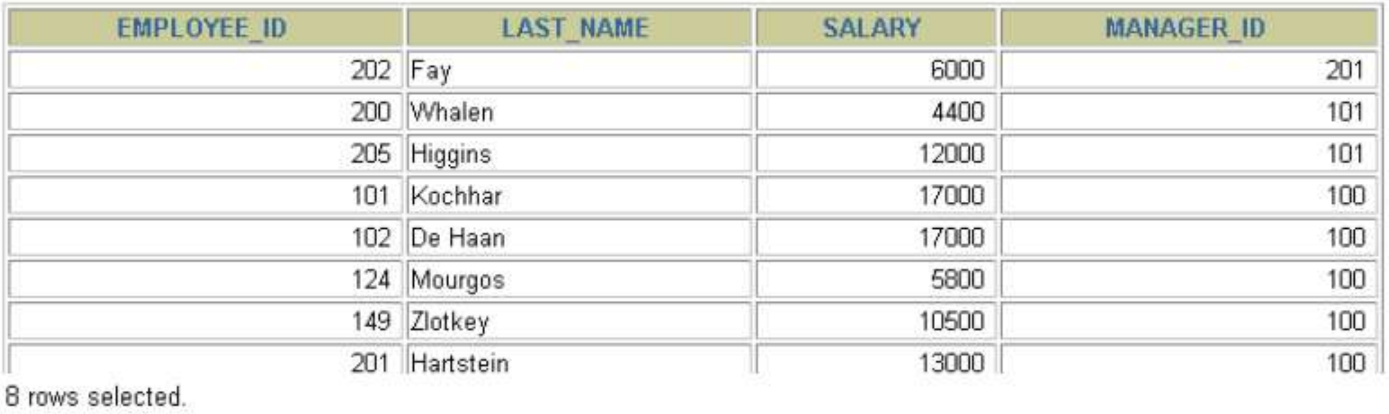
ProductID	ProductName	VendorID	ProductPrice
2X2	Easy Boot	MK	70
3X3	Cosy Sock	MK	15
4X4	Dura Boot	PG	90

Ex3:

- Use the IN membership condition to test for values in a list:



```
SELECT employee_id, last_name, salary, manager_id
FROM   employees
WHERE  manager_id IN (100, 101, 201) ;
```

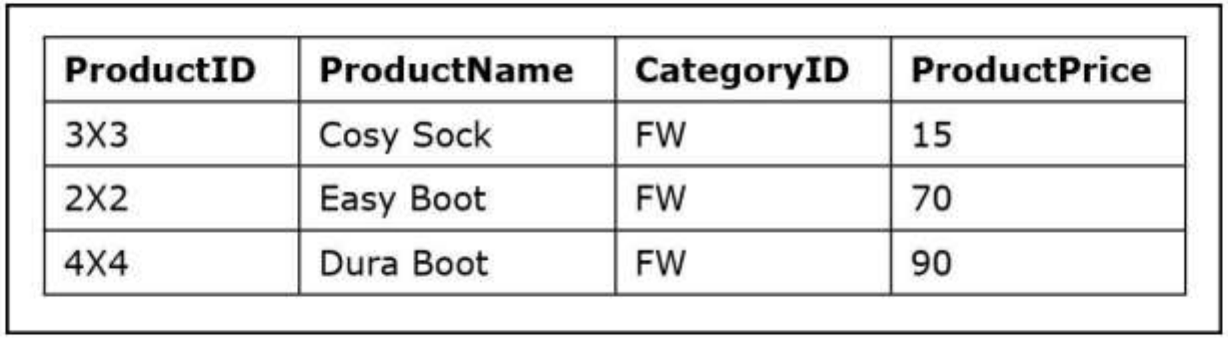


EMPLOYEE_ID	LAST_NAME	SALARY	MANAGER_ID
202	Fay	6000	201
200	Whalen	4400	101
205	Higgins	12000	101
101	Kochhar	17000	100
102	De Haan	17000	100
124	Mourgos	5800	100
149	Zlotkey	10500	100
201	Hartstein	13000	100

8 rows selected.

Order By: (low to high)

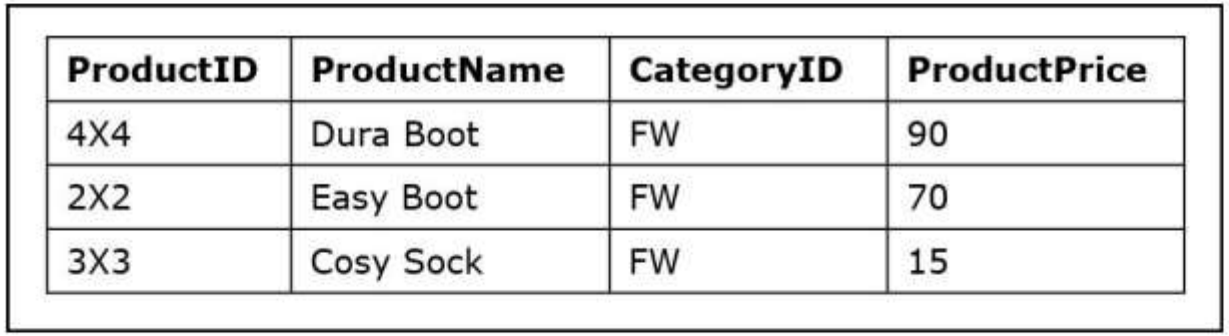
```
SELECT        productid, productname, categoryid,
              productprice
FROM          product
WHERE         categoryid = 'FW'
ORDER BY      productprice;
```



ProductID	ProductName	CategoryID	ProductPrice
3X3	Cosy Sock	FW	15
2X2	Easy Boot	FW	70
4X4	Dura Boot	FW	90

Order By: Desc (high to low)

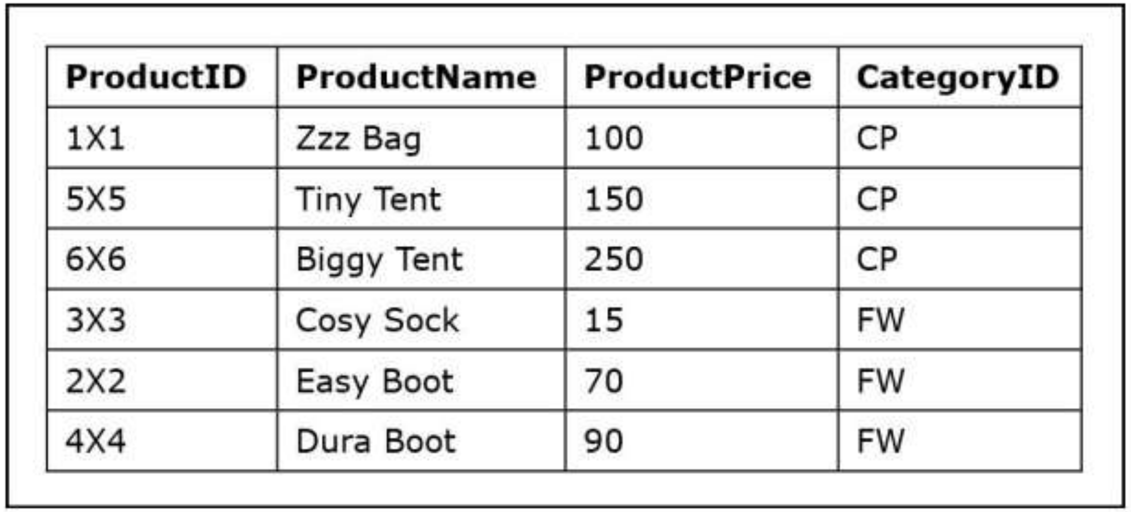
```
SELECT        productid, productname, categoryid,
              productprice
FROM          product
WHERE         categoryid = 'FW'
ORDER BY      productprice DESC;
```



ProductID	ProductName	CategoryID	ProductPrice
4X4	Dura Boot	FW	90
2X2	Easy Boot	FW	70
3X3	Cosy Sock	FW	15

Order By: multi-column

```
SELECT        productid, productname, categoryid,
              productprice
FROM          product
ORDER BY      categoryid, productprice;
```



ProductID	ProductName	ProductPrice	CategoryID
1X1	Zzz Bag	100	CP
5X5	Tiny Tent	150	CP
6X6	Biggy Tent	250	CP
3X3	Cosy Sock	15	FW
2X2	Easy Boot	70	FW
4X4	Dura Boot	90	FW

UPDATE

Insert Statement 1:

```
INSERT INTO product VALUES ('7×7','Airy Sock',1000,'MK','CP');
```

Update Statement 1:

```
UPDATE          product
SET             productprice = 10
WHERE           productid = '7×7';
```

Alter Statement 3:

```
ALTER TABLE product ADD
(discount NUMERIC(3,2) );
```

Update Statement 2:

```
UPDATE product
SET discount = 0.2;
```

Update Statement 3:

```
UPDATE product
SET discount = 0.3
WHERE vendorid = 'MK';
```

Alter Statement 4:

```
ALTER TABLE product DROP (discount);
```

Thanks