TEKNOTURF

RDBMS CONCEPTS

IN THIS MODULE YOU WILL LEARN

TEKNOTURF

- Information System
- File Based System
- Database Management System
- Database Users
- Data Model
- Properties of Relations
- RDBMS
- Structured Query Language
- Standard and best Practices of SQL
- Components of SQL

INFORMATION SYSTEM

TEKNOTURF

An Information system is an organized collection of hardware, software, supplies, policies, procedures and people, which stores process and provides access to information.

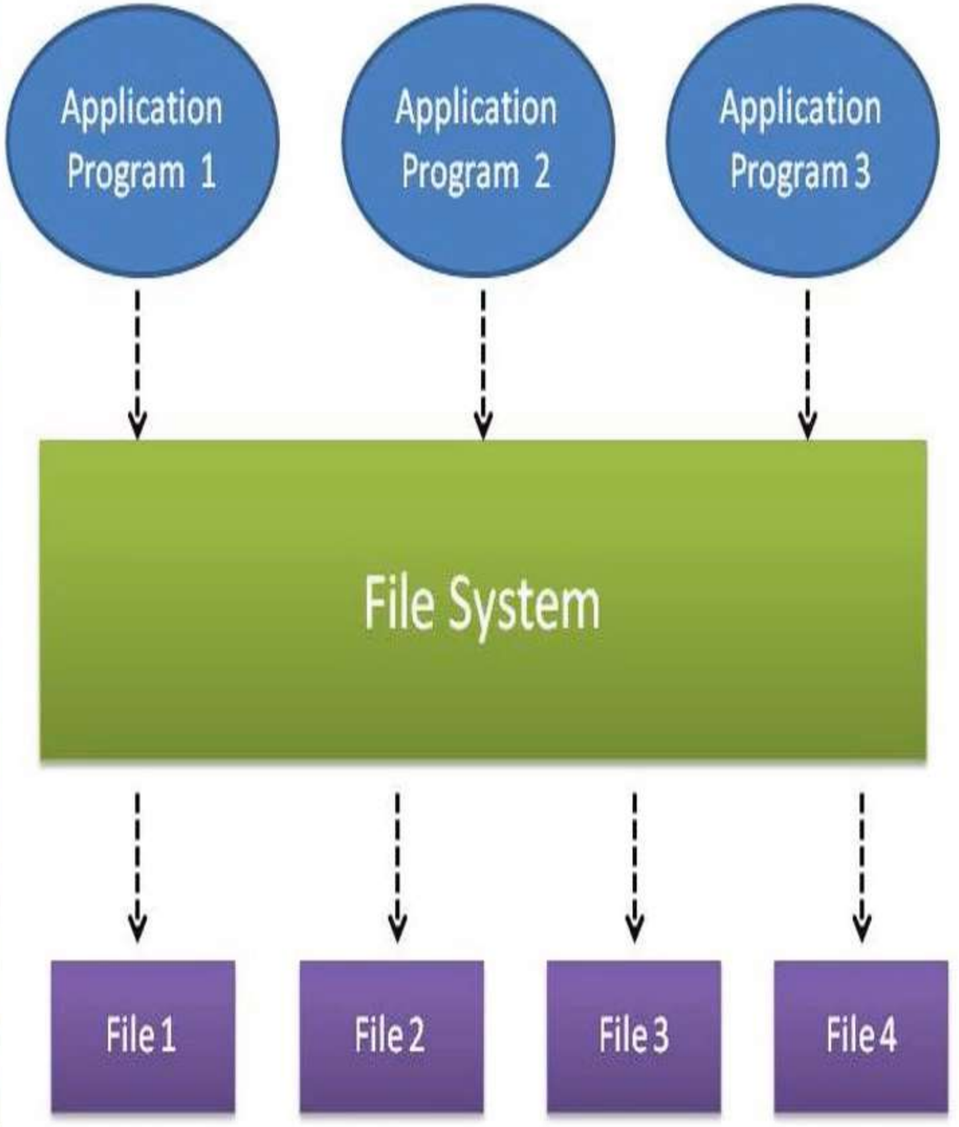
TRADITIONAL APPROACH - FILE BASED SYSTEM

TEKNOTURF

Information is stored in flat files, which are maintained by the file system under the operating systems control.

Application programs go through the file system in order to access these flat files.

Records consist of various fields, which are delimited by a space, comma, pipe, or any special character etc.,



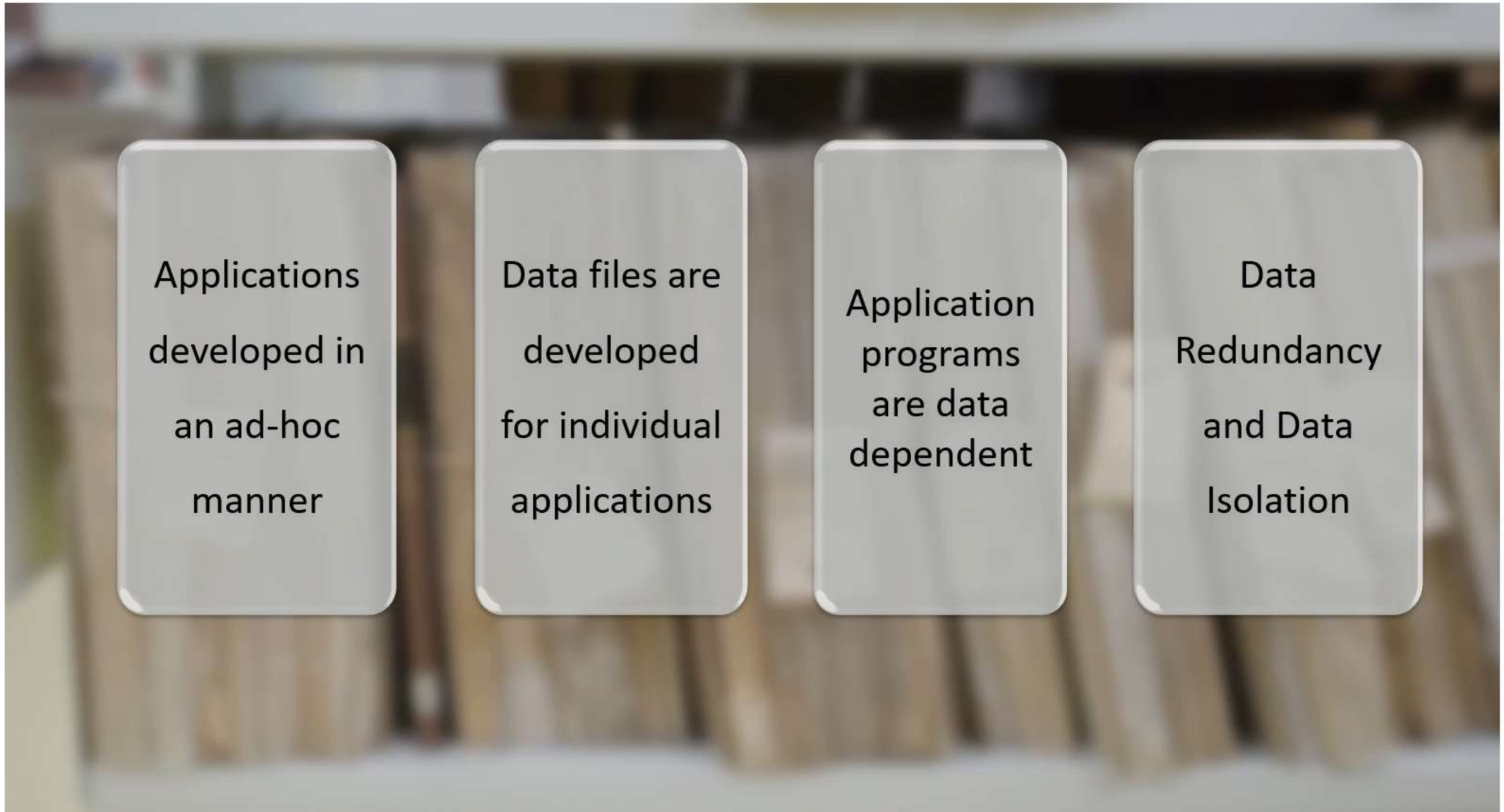
TRADITIONAL APPROACH - FILE BASED SYSTEM

This is really great. We are able to free up all that space by moving all the data on the computer.

It's an improvement. We don't have to search for a file in the filing room. But it's still difficult to produce reports across sales, product and customer data, because they are maintained on a separate file systems.

TRADITIONAL APPROACH - DISADVANTAGES

TEKNOTURF



DATABASE MANAGEMENT SYSTEM - INTRODUCTION

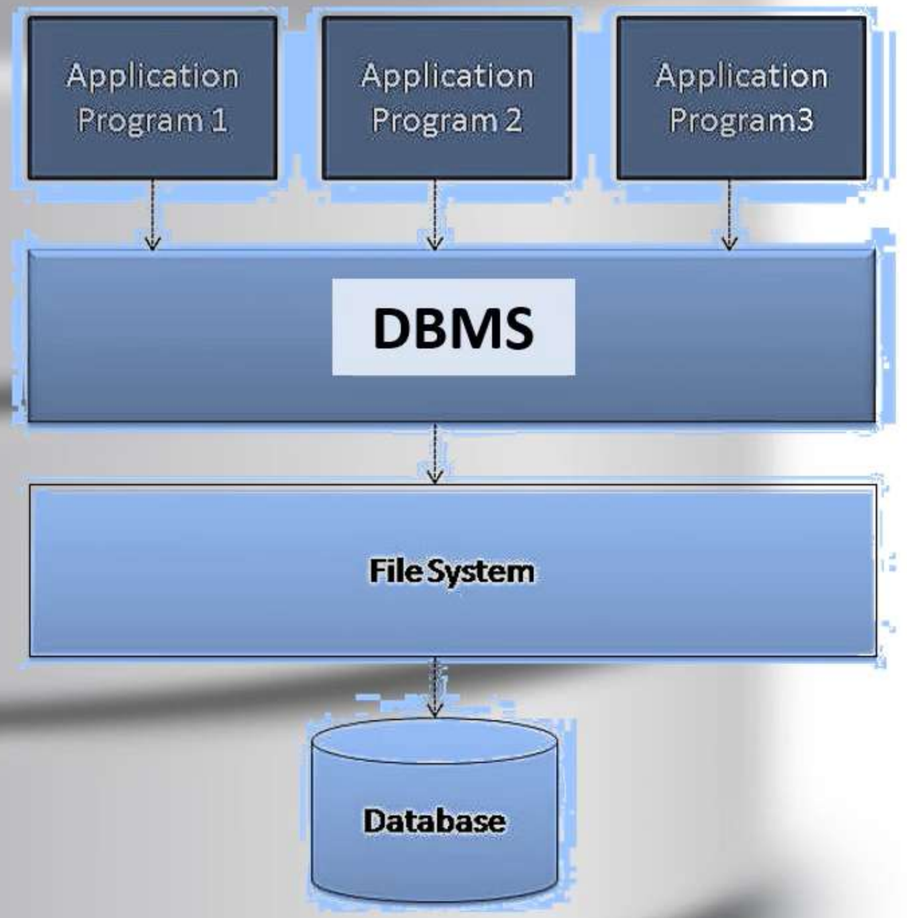
TEKNOTURF

DATABASE

A shared collection of logically related Data (and a description of this Data), designed to meet the needs of an organization.

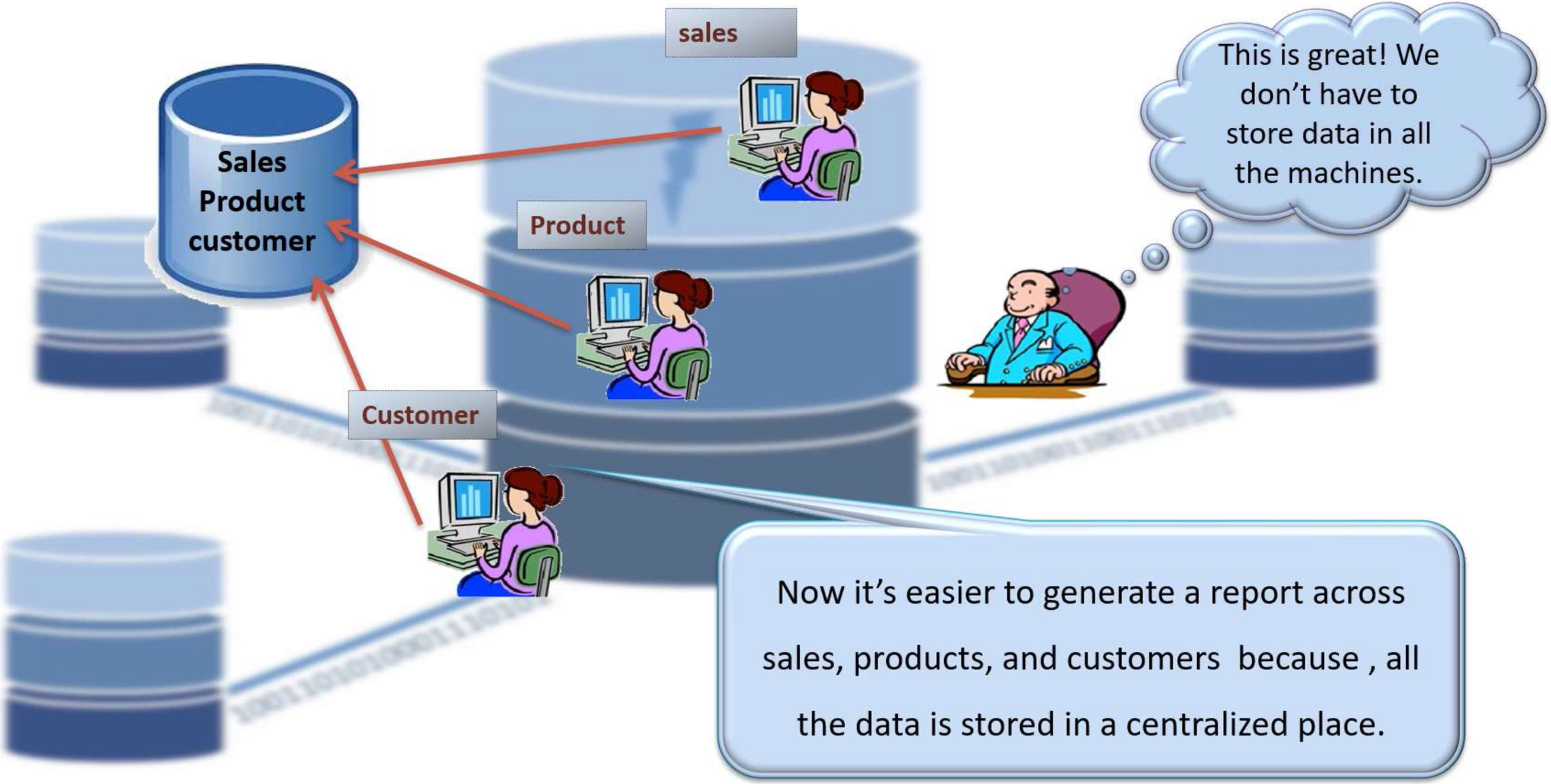
Database Management System (DBMS)

Software that enable users to define, create and maintain the Database and provides controlled access to the Database.



DATABASE MANAGEMENT SYSTEM

TEKNOTURF



DATABASE APPROACH

- Centralization of Information Management.
- Data shared by different group of users and application programs.
- Representation of complex relationship. between data.
- Integrity Constraint handling.
- Advanced facilities for backup and recovery.

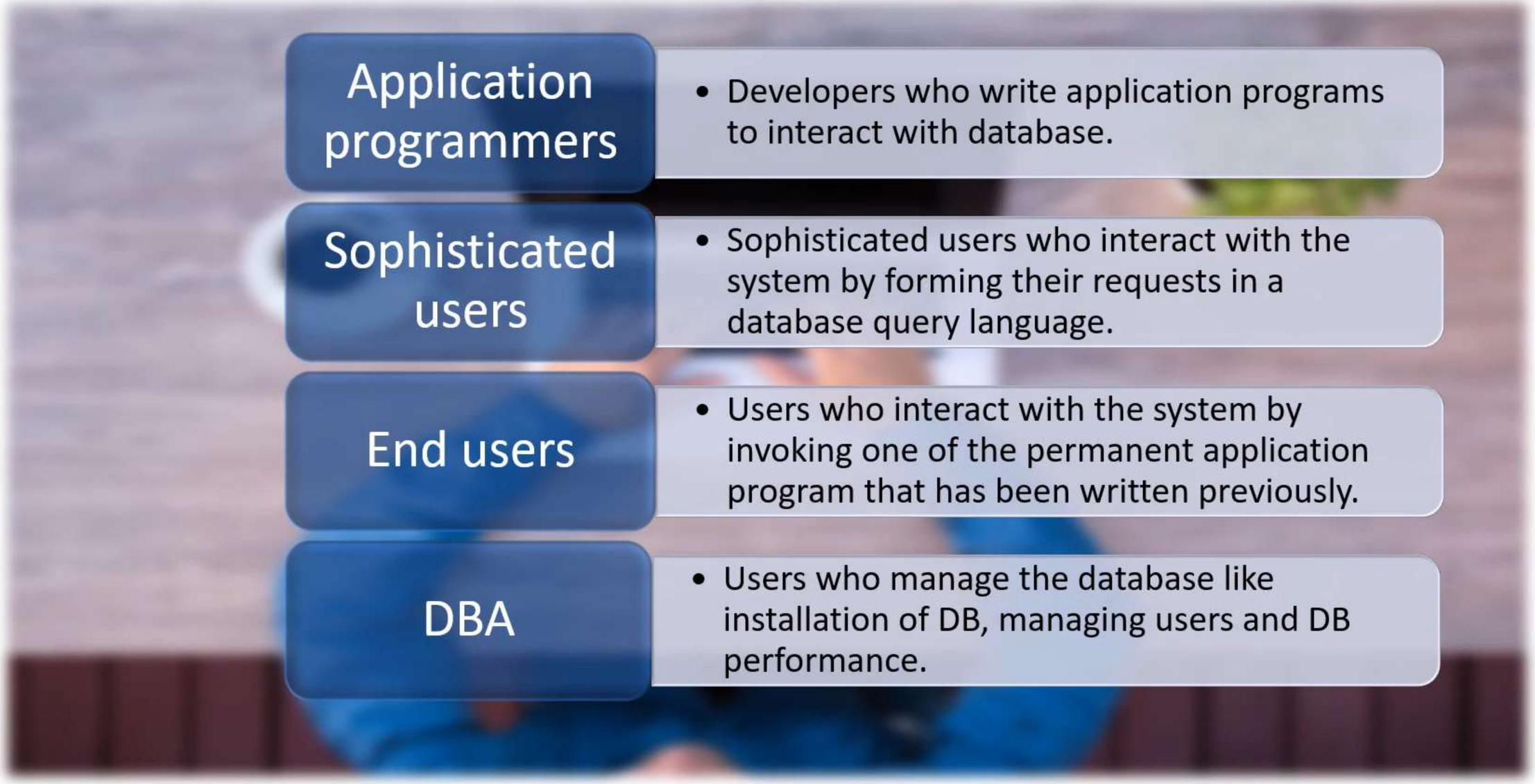
DATABASE MANAGEMENT SYSTEM - ADVANTAGES

TEKNOTURF

- Sharing of data.
- Enforcement of security.
- Enforcement of development and maintaining standards.
- Reduction of redundancy.
- Avoidance of inconsistency across files.
- Maintenance of integrity.
- Data independence.

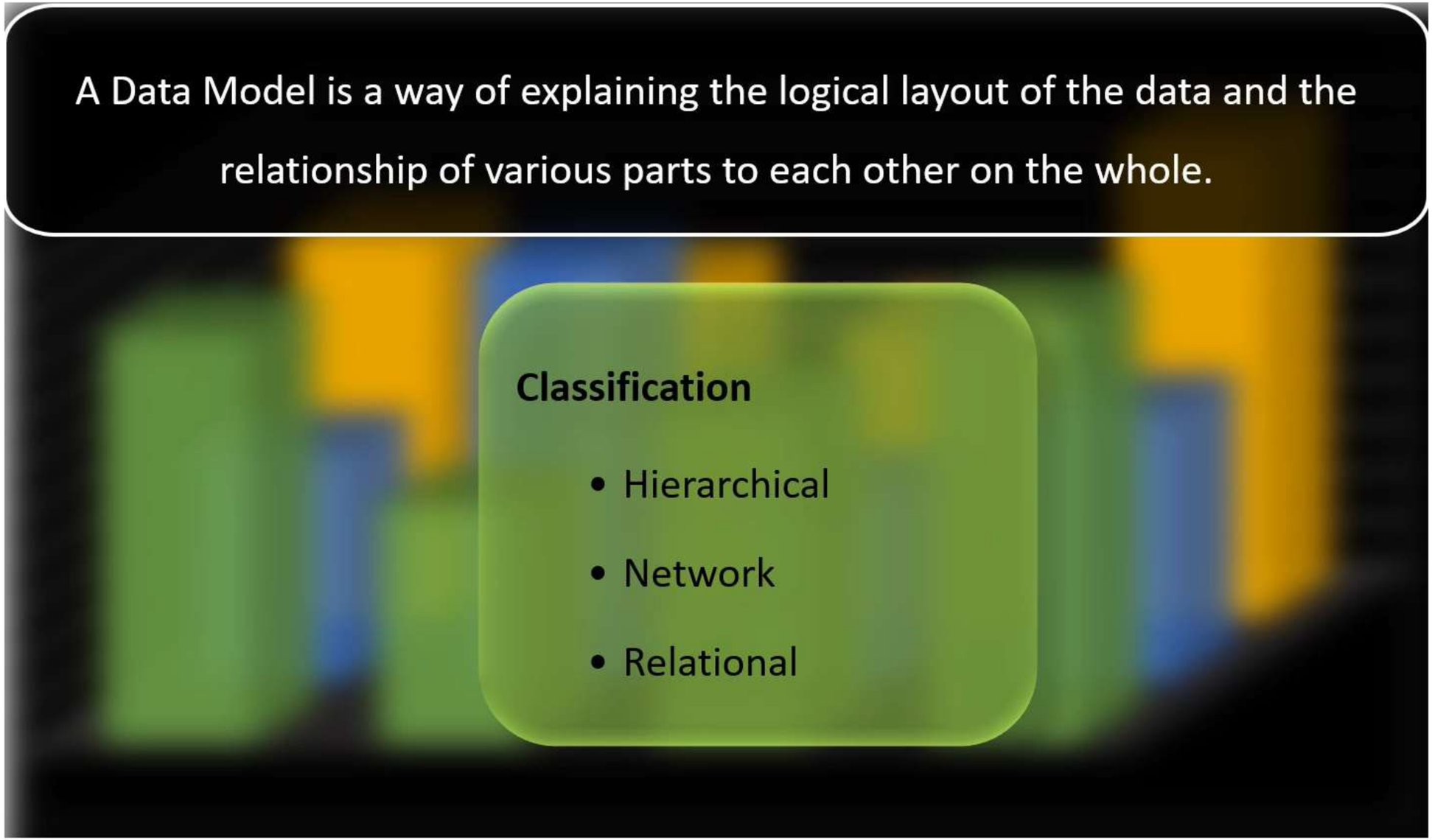
DATABASE USERS

TEKNOTURF

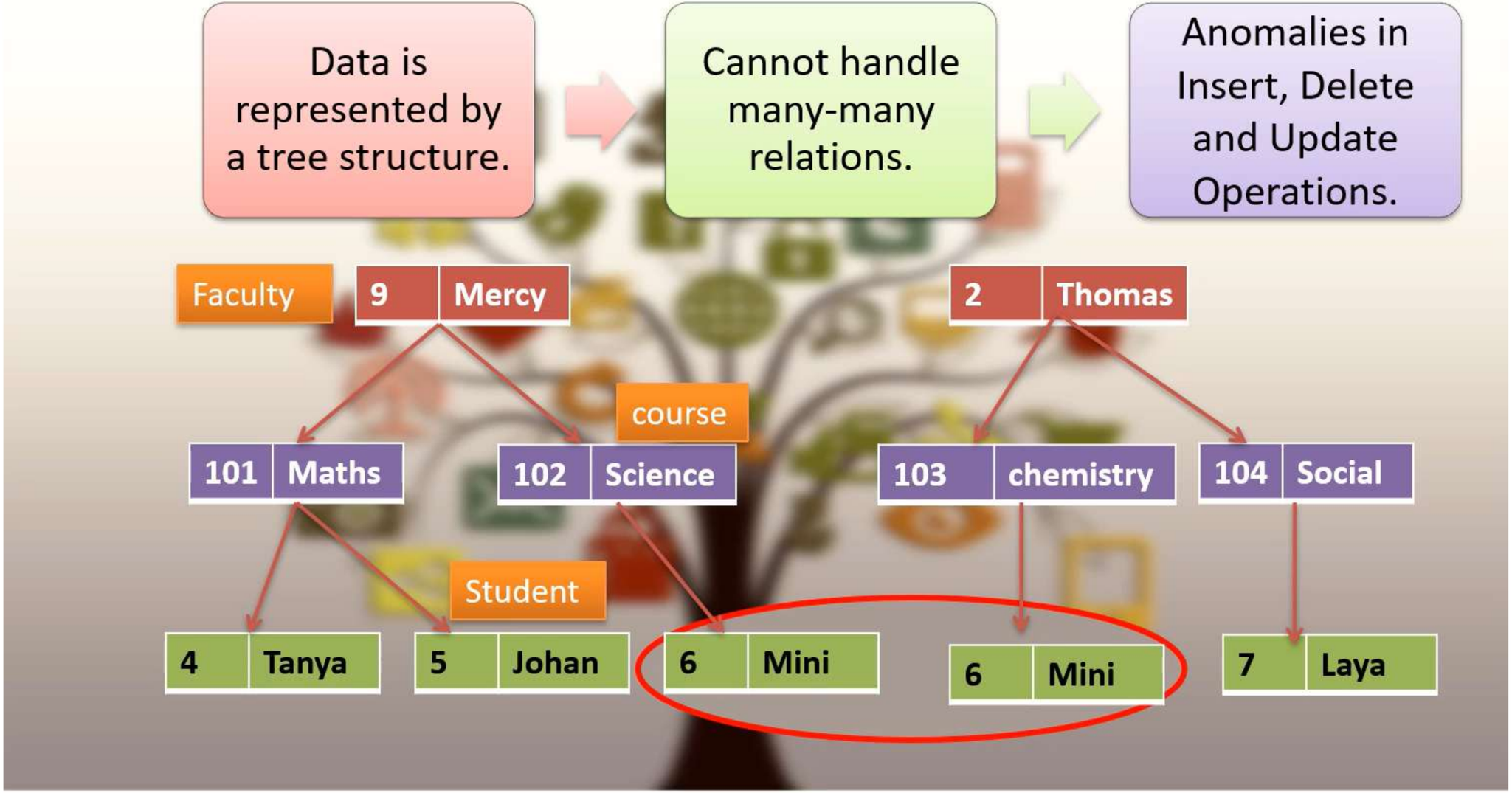


Application programmers	• Developers who write application programs to interact with database.
Sophisticated users	• Sophisticated users who interact with the system by forming their requests in a database query language.
End users	• Users who interact with the system by invoking one of the permanent application program that has been written previously.
DBA	• Users who manage the database like installation of DB, managing users and DB performance.

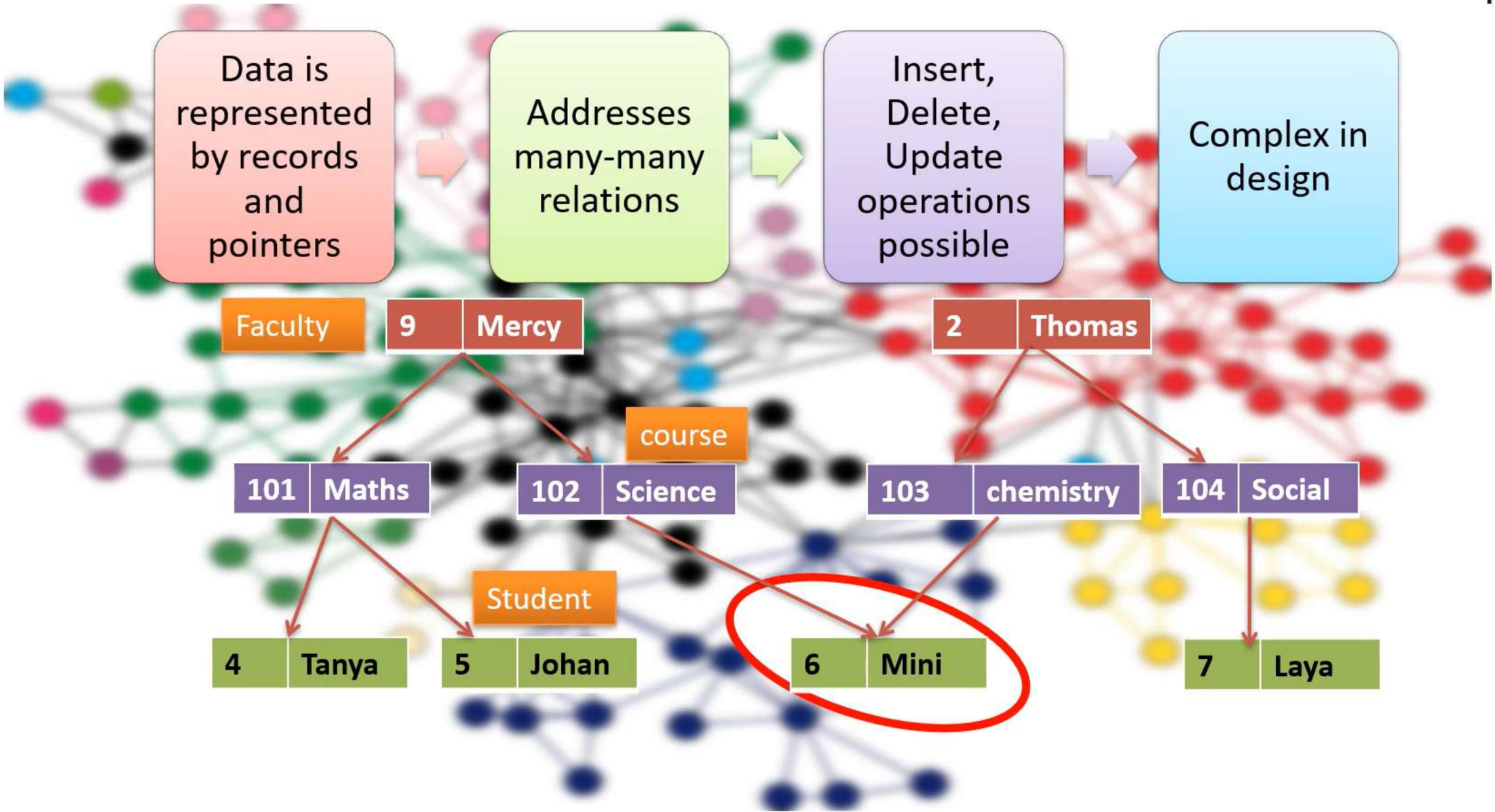
DATA MODEL


A Data Model is a way of explaining the logical layout of the data and the relationship of various parts to each other on the whole.
Classification
• Hierarchical
• Network
• Relational

HIERARCHICAL DATA MODEL



NETWORK DATA MODEL



RELATIONAL DATA MODEL

TEKNOTURF

Relational Database Management system, where the data is kept in tables or relations.

More flexible & easy to use.

Almost any item of the data can be accessed more quickly than the other models.

FACULTYID	FACULTYNAME
9	Mercy
2	Thomas

COURSEID	COURSENAME
101	C Programming
102	Computer Networks
103	RDBMD
104	HTML5

RELATIONSHIP BETWEEN TABLES

TEKNOTURF

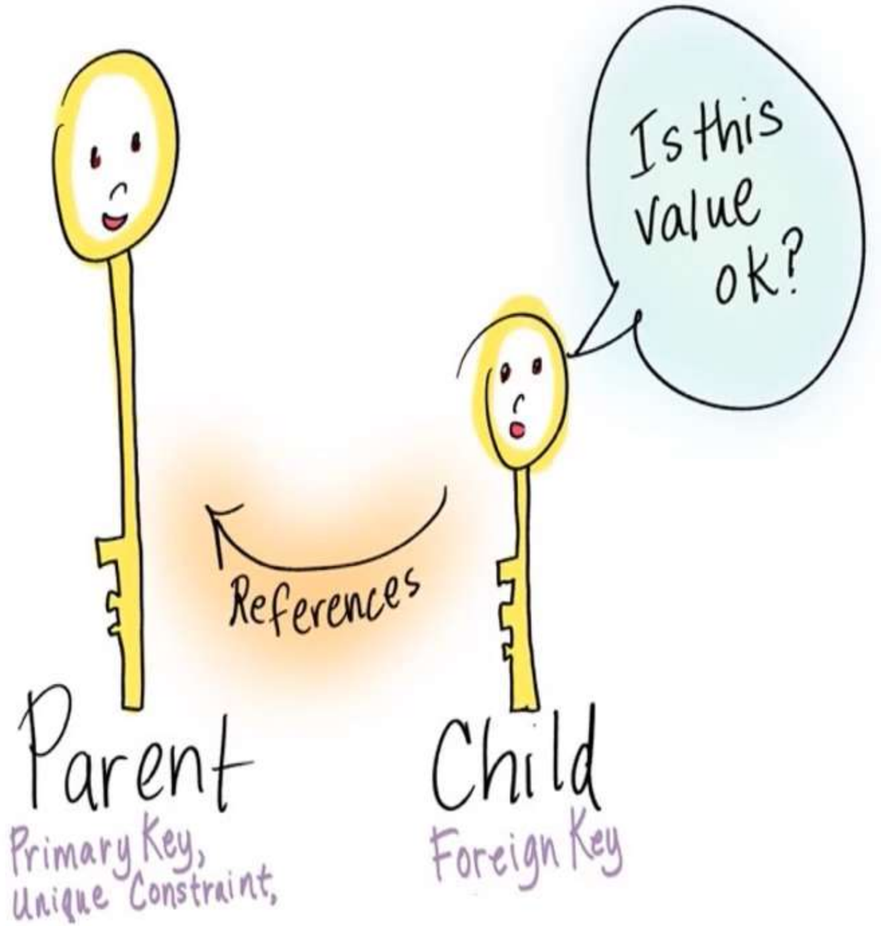
FACULTYID	FACULTYNAME
9	Mercy
2	Thomas

Primary Key

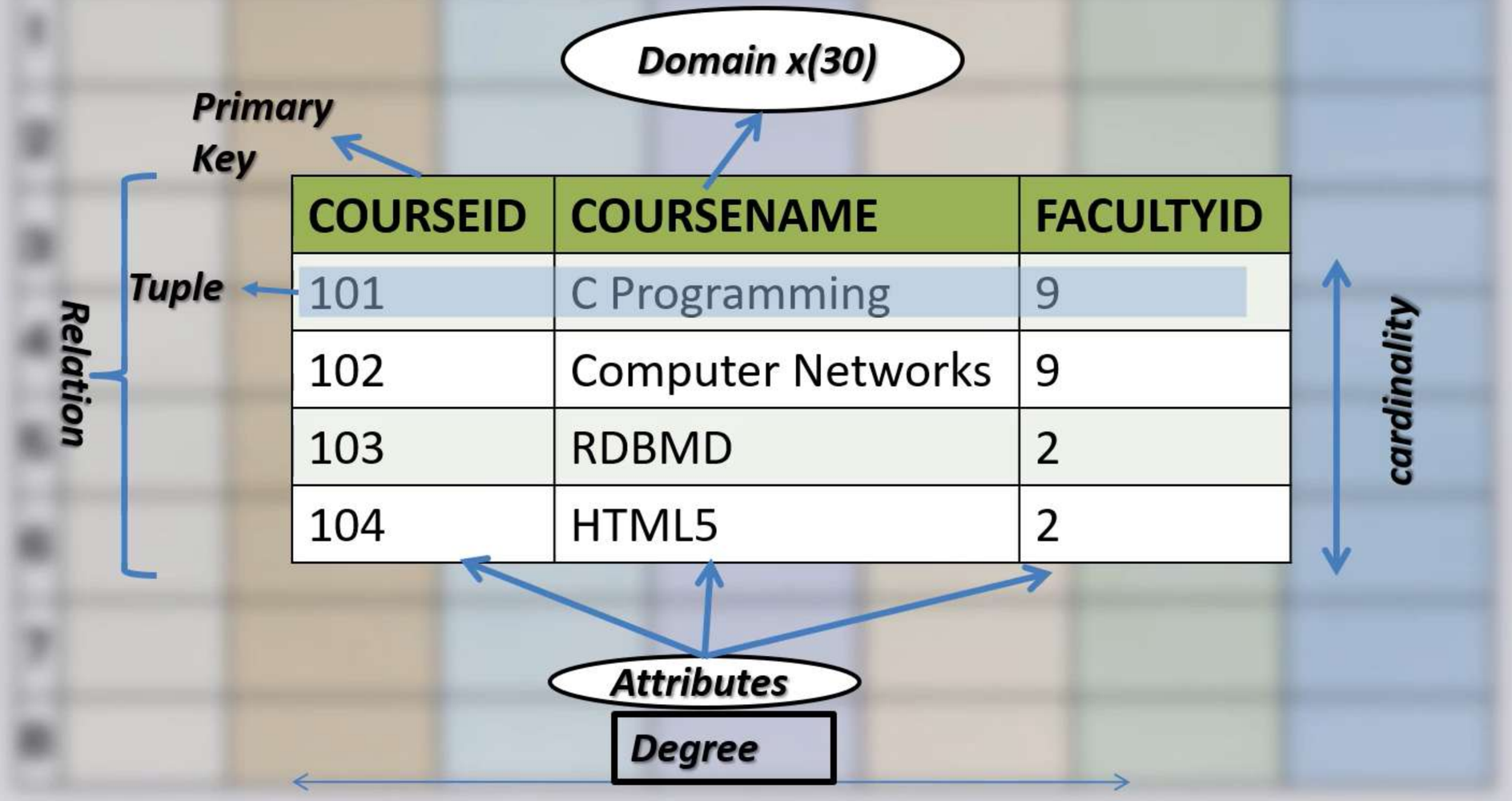
Foreign Key

COURSEID	COURSENAME	FACULTYID
101	C Programming	9
102	Computer Networks	9
103	RDBMD	2
104	HTML5	2

Primary Key



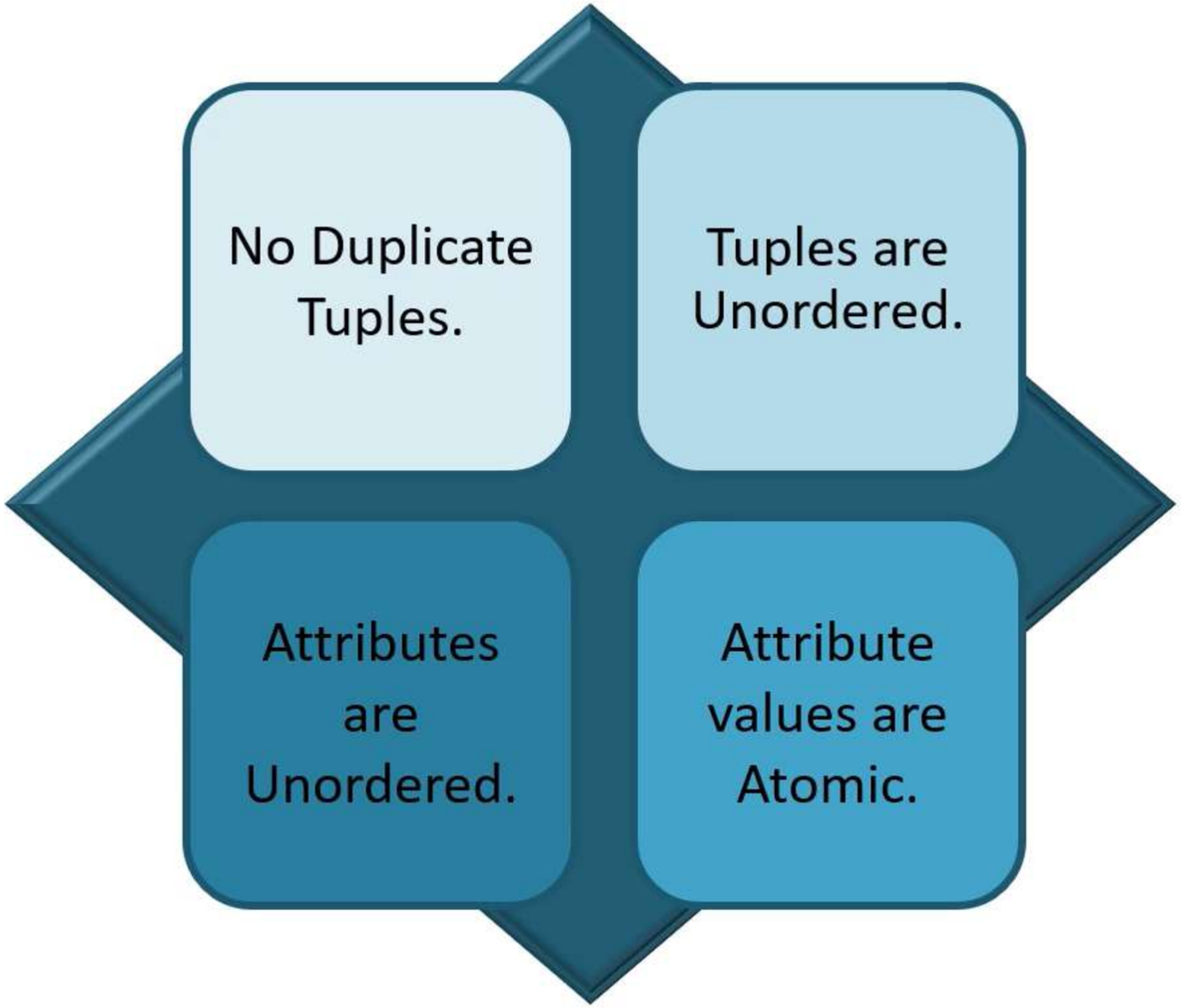
RELATIONAL DATA MODEL - TERMINOLOGY



COURSEID	COURSENAME	FACULTYID
101	C Programming	9
102	Computer Networks	9
103	RDBMD	2
104	HTML5	2

PROPERTIES OF RELATIONS

TEKNOTURF



RELATIONAL DATABASE MANAGEMENT SYSTEM

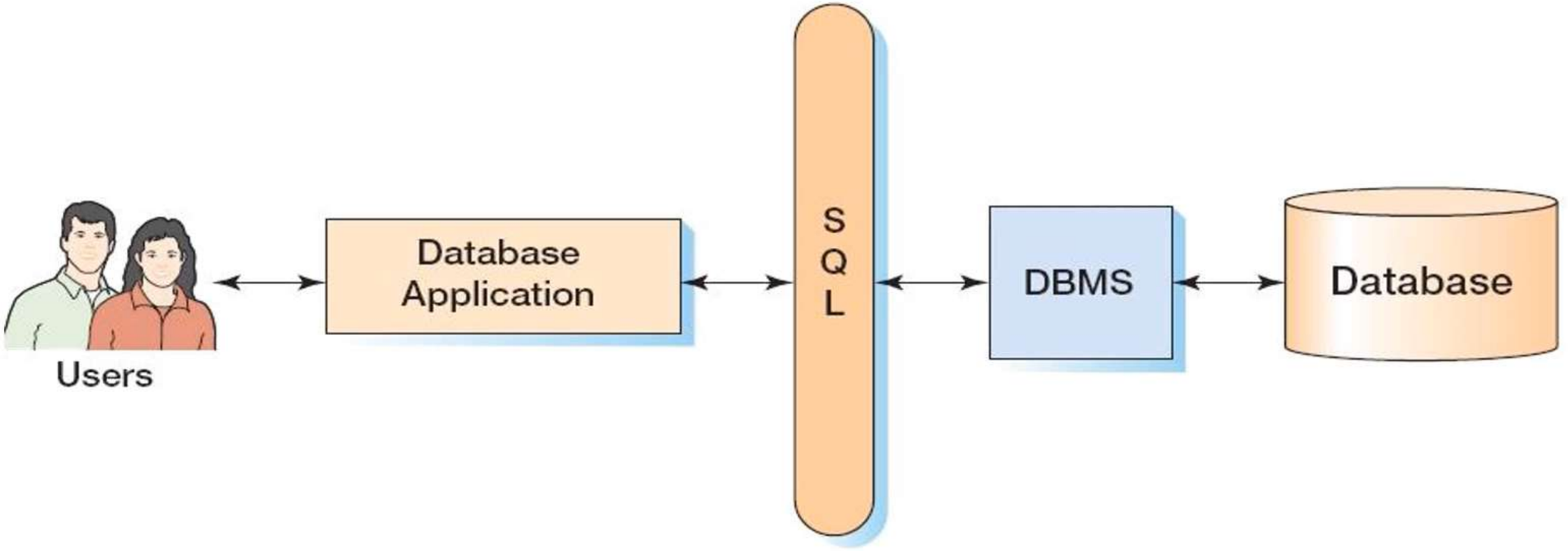
TEKNOTURF

A Relational Database Management System (RDBMS) is a system software, that let's you Create, Update and Administer a Relational Database.

RDBMSs uses Structured Query Language (SQL) to access the Database.

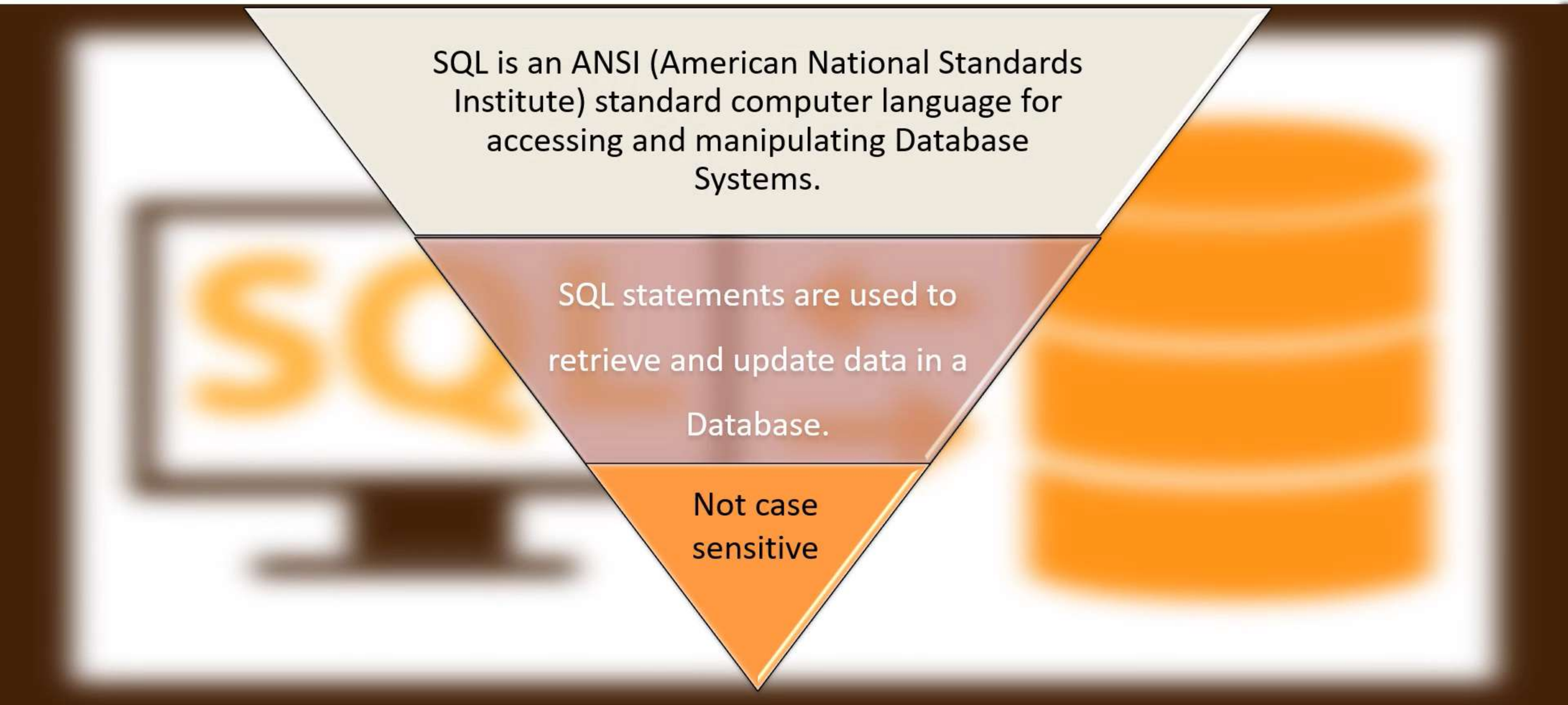
RDBMS Products are Oracle, DB2 and MYSQL etc.,

STRUCTURED QUERY LANGUAGE



STRUCTURED QUERY LANGUAGE

TEKNOTURF



STANDARD AND BEST PRACTICES OF SQL

TEKNOTURF

Object Naming Conversions

- Use Pascal notation;
- Examples: Products, Customers.

Column Names:

- Use the singular form of nouns
- FirstName, address.

Each table must have a primary key

Use upper case for all SQL keywords

- SELECT, INSERT, UPDATE, WHERE, AND, OR, LIKE, etc.,

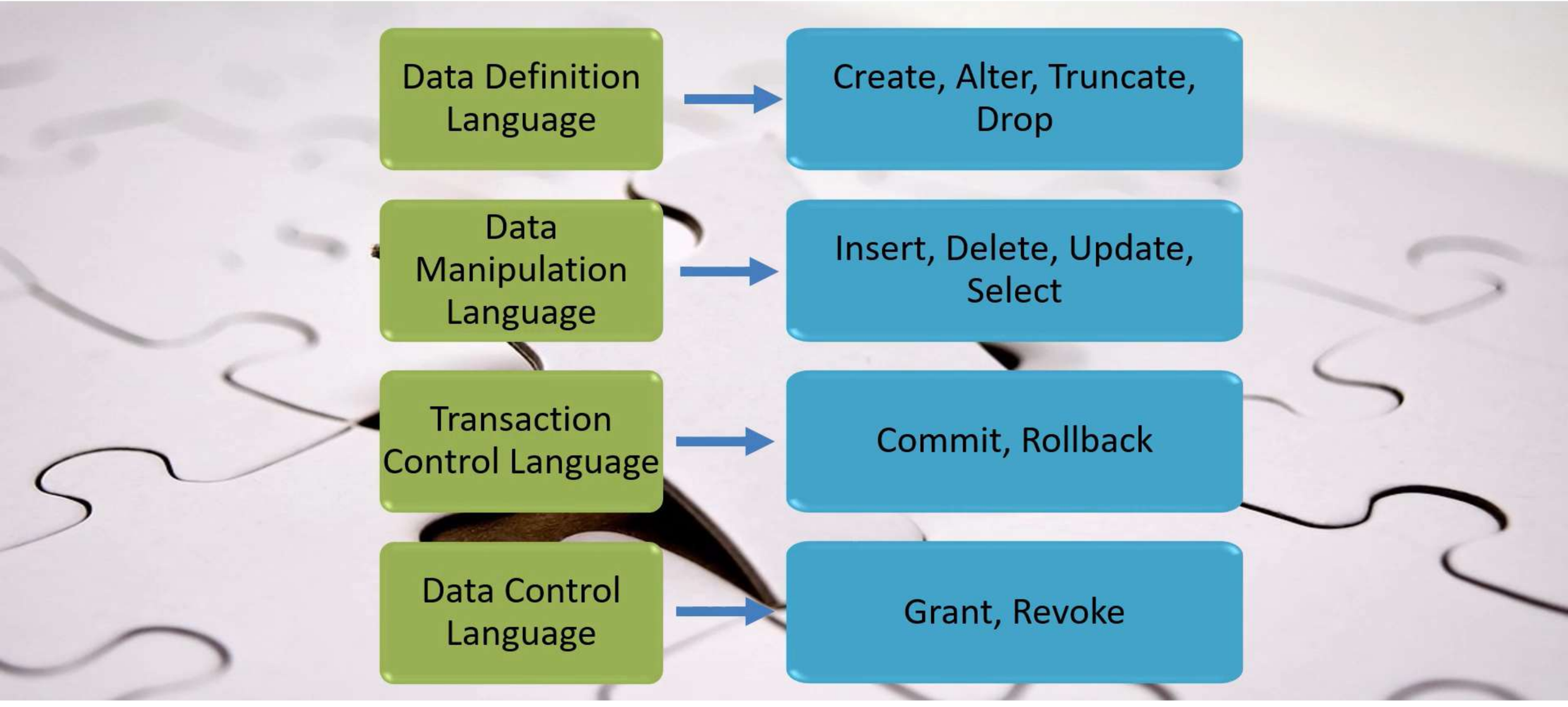
STANDARD AND BEST PRACTICES OF SQL

TEKNOTURF

- Do not use white space in identifiers.
  - Example : firstname, lastname
- Use parentheses to increase readability
  - WHERE (color='red' AND (size = 1 OR size = 2))
- Indent code to improve readability.
- Use ANSI Joins instead of old style joins.
- Do not use SELECT *
- Always use table aliases when your SQL statement involves more than one table.
- Do not use column numbers in the ORDER BY clause.
- Always use a column list in INSERT statements.

COMPONENTS OF SQL

TEKNOTURF



DATA DEFINITION LANGUAGE

TEKNOTURF

All DB objects like TABLE, VIEW, SEQUENCE, INDEX are created using '**CREATE**' statement of DDL.

Modification of Object's structure is done using '**ALTER**'.

Removing of data or structure are done using '**TRUNCATE**' and '**DROP**' respectively.

DDL statements are **AUTO COMMIT**

Summary

- Information system.
- File based system.
- Database management system.
- Database users.
- Data model.
- Properties of relations.
- RDBMS.
- Structured query language.
- Standard and best Practices of SQL.
- Components of SQL.

Description Edit Grading view

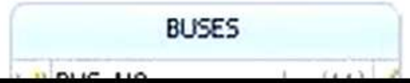
Create Users table

Requested files: dd13.sql
(Download)

Type of work: Individual work

Refer the below schema and create the table USERS.

Column Name	Datatype	Size	Constraint	Constraint name
User_id	Number	11	Primary key	PK_USERS
Name	Varchar2	20		
Address	Varchar2	100		
Phno	Number	11		
Emailid	Varchar2	30		



Description | Edit | Grading view

File List | Save | Compile & Run | Evaluate

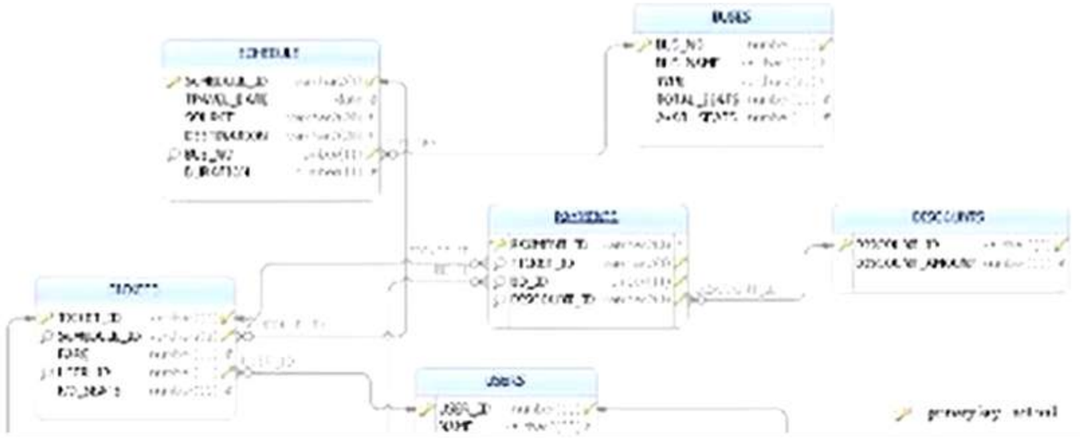
dd13.sql

```
1  create table users(
2      user_id varchar2(11),
3      name varchar2(20),
4      address varchar2(100),
5      phno number(11),
6      eMaIliD varchar2(30),
7      constraint pk primary key(user_id)
8      );
```

Question Description

Refer the below schema and create the table USERS.

Column Name	Datatype	Size	Constraint	Constraint name
User_id	Number	11	Primary key	PK_USERS
Name	Varchar2	20		
Address	Varchar2	100		
Phno	Number	11		
Emailid	Varchar2	30		



Description Edit Grading view

File List | Save | Compile & Run | Evaluate | Full screen | Description

Run (Ctrl-F11)

dd13.sql

```
1  create table users(
2      user_id varchar2(11),
3      name varchar2(20),
4      address varchar2(100),
5      phno number(11),
6      eMaIliD varchar2(30),
7      constraint pk primary key(user_id)
8      );
```

rms/edit.php?id=13209&useri...

Description

>_ Console: connection closed (Running: 1 seg)

```
Table created.
```

File List

dd13.sql

```
1  create ta
2      user_
3      name
4      addre
5      phno
6      eMaIl
7      const
8      );
```

05:12

Description Edit Grading view

File List | Save | Compile & Run | Evaluate | Full screen | Description

Evaluate (Shift-F11)

dd13.sql

```
1  create table users(
2      user_id varchar2(11),
3      name varchar2(20),
4      address varchar2(100),
5      phno number(11),
6      eMaIliD varchar2(30),
7      constraint pk primary key(user_id)
8      );
```

rms/edit.php?id=13209&useri...

05:27

Description Edit Grading view

File List | Save | Compile & Run | Evaluate | Full screen | Description

dd13.sql

```
1  create table users(
2      user_id varchar2(11),
3      name varchar2(20),
4      address varchar2(100),
5      phno number(11),
6      eMaIliD varchar2(30),
7      constraint pk primary key(user_id)
8      );
```

Proposed grade: 0 / 100

Result Description

Test case - Column name and datatype check

Check the datatype(s) for the co

USER_ID

Test case - Column constraint name check

Check the constraint(s) name for

USER_ID
>

Description Edit Grading view

File List | Save | Compile & Run | Evaluate | Full screen | Description

dd13.sql

```
1  create table users(
2      user_id number(11),
3      name varchar2(20),
4      address varchar2(100),
5      phno number(11),
6      eMaIliD varchar2(30),
7      constraint pk_users primary key(user_id)
8      );
```

Proposed grade: 75 / 100

Result Description

Test case - Column constraint name check

Check the constraint(s) name for th

USER_ID

Summary of tests

```
+----------------------------+
| 3 tests run / 2 test passed |
+----------------------------+
```

Description Edit Grading view

File List | Save | Compile & Run | Evaluate | Full screen | Description

dd13.sql

```
1  create table users(
2      user_id number(11),
3      name varchar2(20),
4      address varchar2(100),
5      phno number(11),
6      eMaIliD varchar2(30),
7      constraint pk_users primary key(user_id)
8      );
```

Proposed grade: 75 / 100

Result Description

Test case - Column constraint name check

Check the constraint(s) name for th

USER_ID

Summary of tests

```
+-----------------------------+
| 3 tests run / 2 test passed |
+-----------------------------+
```

rms/edit.php?id=13209&useri...

07:22

Description Edit Grading view

File List | Save | Compile & Run | Evaluate | Full screen | Description

dd13.sql

```
1  create table users(
2      user_id number(11),
3      name varchar2(20),
4      address varchar2(100),
5      phno number(11),
6      eMaIliD varchar2(30),
7      constraint pk_users primary key(user_id)
8      );
```

Proposed grade: 100 / 100

Result Description

Summary of tests

```
+----------------------------+
| 3 tests run / 3 test passed |
+----------------------------+
```

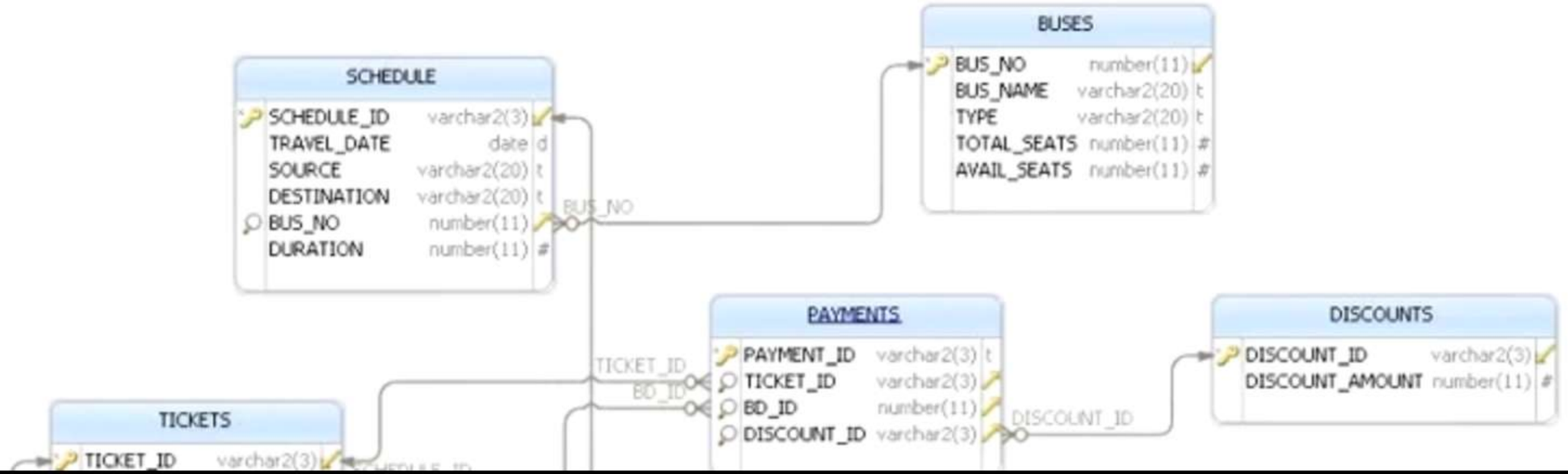
Description Edit Grading view

Add a new column

Requested files: ddl3.sql
(Download)

Type of work: Individual work

Write a query to add a new column, LiveTracker of data type char(1). This will be used to track the live location of the bus, for the journey.



Description Edit Grading view

File List | Save | Compile & Run | Evaluate | Full screen

Run (Ctrl-F11)

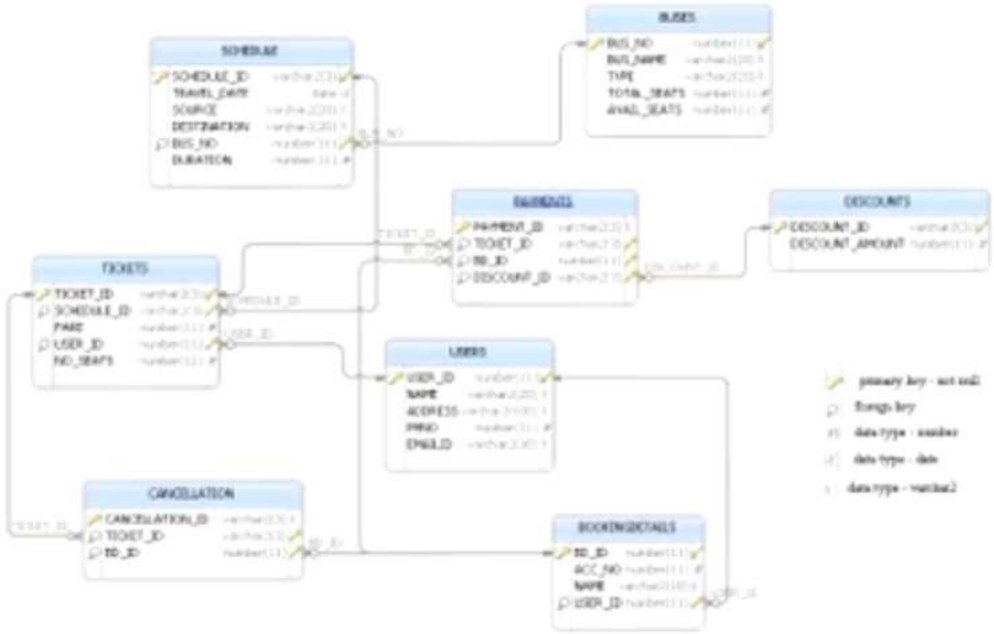
ddl3.sql

```
1 desc buses;
```

forms/edit.php?id=13214&userid=3303#

Question Description

Write a query to add a new column, LiveTracker of data type char(1). This will be used to track the live location of the bus, for the journey.



02:23

Description Edit Grading view

File List Save

ddl3.sql

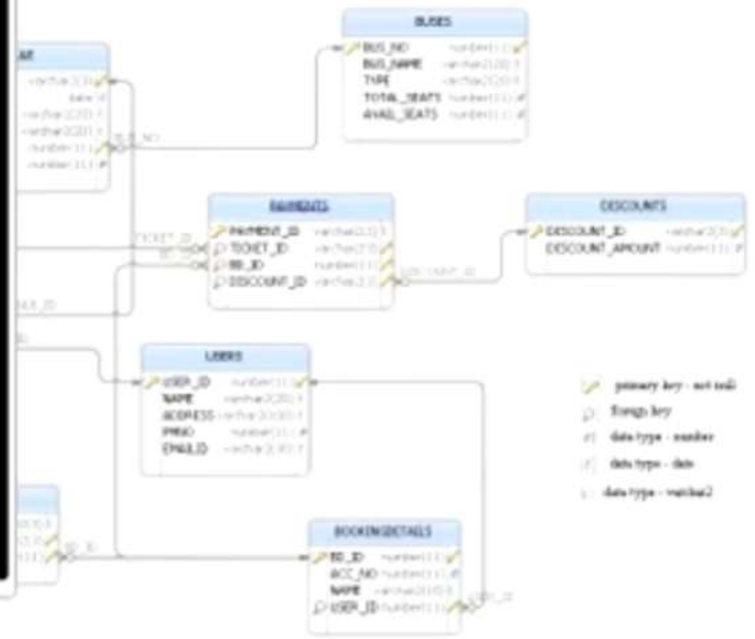
```
1 desc buses;
```

Question Description

dd a new column, LiveTracker of data type char(1).
o track the live location of the bus, for the

>_ Console: connection closed (Running)

```
Name                                      Null?    Type
----------------------------------------- -------- ----------------------------
BUS_NO                                    NOT NULL NUMBER(11)
BUS_NAME                                           VARCHAR2(20)
TYPE                                               VARCHAR2(20)
TOTAL_SEATS                                        NUMBER(11)
AVAIL_SEATS                                        NUMBER(11)
```



02:33

Description Edit Grading view

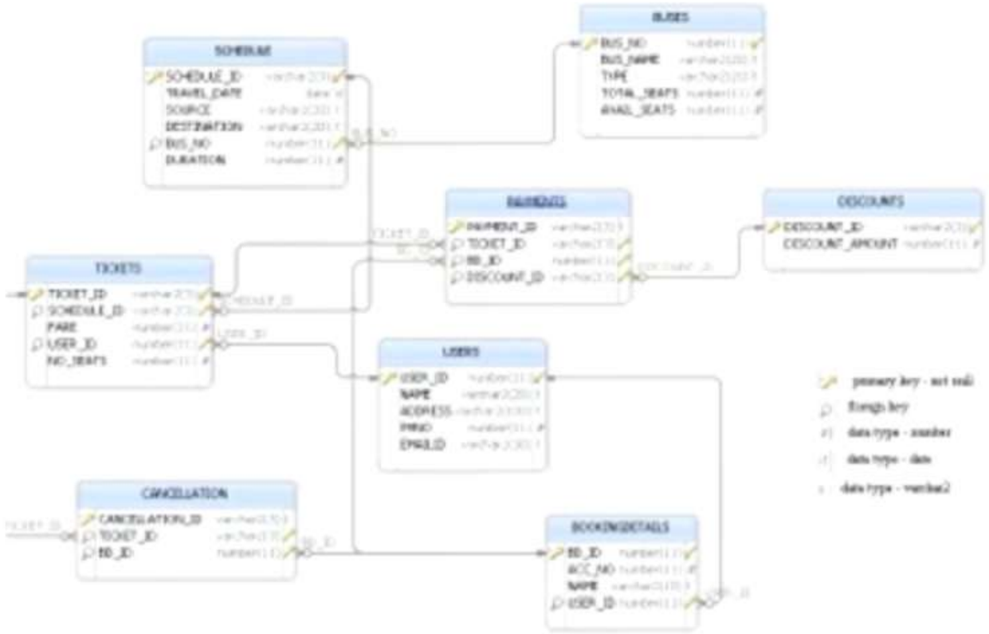
File List Save Compile & Run Evaluate Full screen

ddl3.sql

```
1 alter table buses add livetracker char(1);
```

Question Description

rite a query to add a new column, LiveTracker of data type char(1).
iis will be used to track the live location of the bus, for the
urney.



forms/edit.php?id=13214&useri...

03:33

Description Edit Grading view

File List | Save | Compile & Run | Evaluate | Full screen | Description

ddl3.sql

```
1 alter table buses add livetracker char(1);
```

Proposed grade: 100 / 100

Result Description

Summary of tests

3 tests run / 3 test passed

04:30

TEKNOTURF

CREATING AND MANAGING TABLES

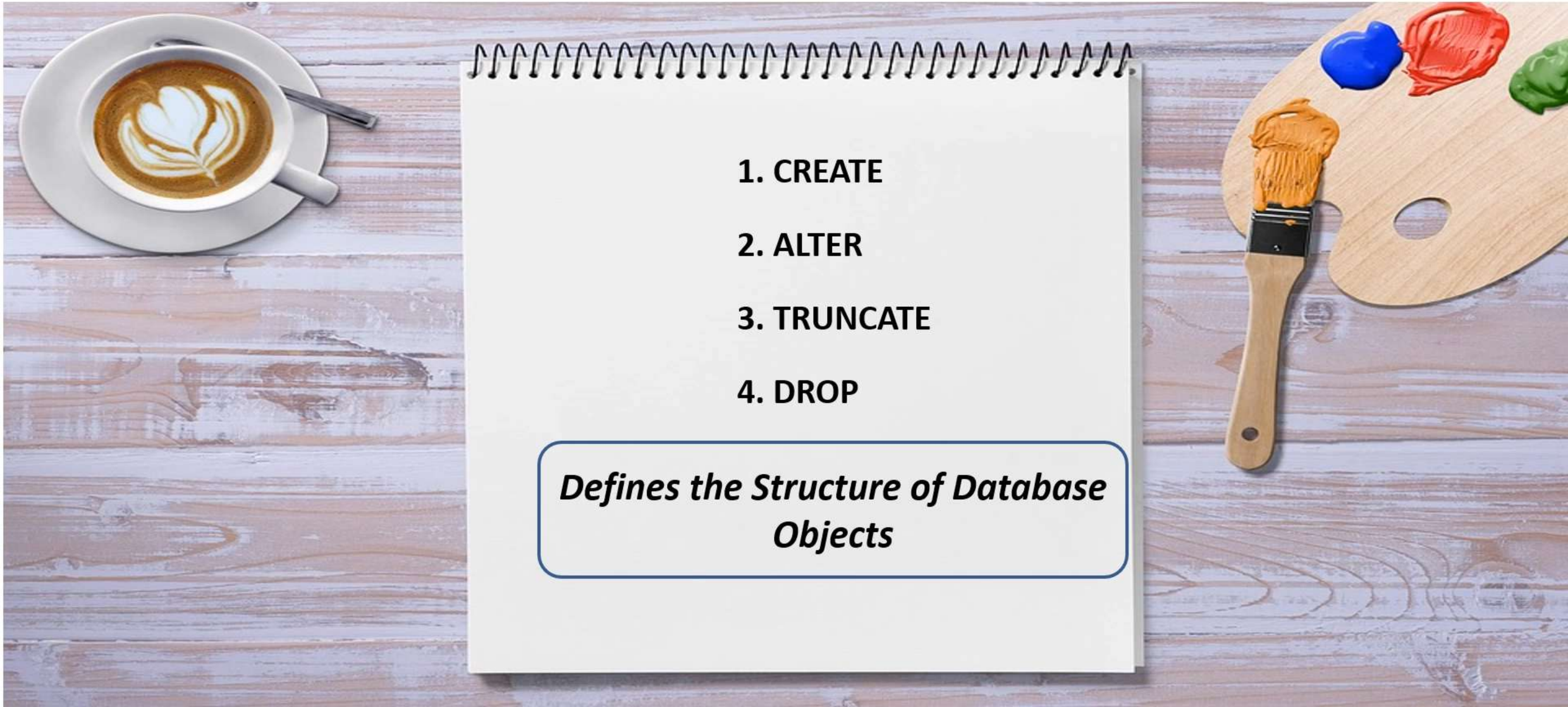
IN THIS MODULE YOU WILL LEARN

TEKNOTURF

- Data types
- Create table
- Alter table
- Drop and Truncate table
- Various constraints

DATA DEFINITION LANGUAGE

TEKNOTURF



1. CREATE
2. ALTER
3. TRUNCATE
4. DROP

Defines the Structure of Database Objects

INSURANCE MANAGEMENT SYSTEM(IMS)

TEKNOTURF

We will be using IMS through out this module

Customer
• CId • CName • Phoneno • Email • Address

Policy
• PId • PName • PPeriodInYears • MinAmountPerMonth

PolicyEnrollment
• EnrollmentId • CId • PId • Amount • DueDate • PaidDate • Penalty

SAMPLE RECORDS

TEKNOTURF

CID	CNAME	PHONENO	DOB	EMAILID	ADDRESS
1	Tom	9876523190	17-MAR-87	tom@gmail.com	chennai
2	John	8765432190	26-JAN-86	john@yahoo.com	delhi
3	Ram	7654321890	14-DEC-85	ram@gmail.com	pune
4	Tiny	9012365478	28-MAY-86	NULL	chennai

Customer

PID	PNAME	PPERIODINYEARS	MINAMOUNTPERMONTH
MBP	Money Back Plan	20	1000
PP	Personal Protect	15	1500

Policy

ENROLLMENTID	CID	PID	DUEDATE	PAIDDATE	AMOUNT	PENALTY
101	3	MBP	12-Dec-2017	11-Dec-2017	2000	0
102	1	PP	15-Mar-2018	13-Mar-2018	3000	0
103	2	PP	15-Feb-2018	22-Feb-2018	4000	200

PolicyEnrollment

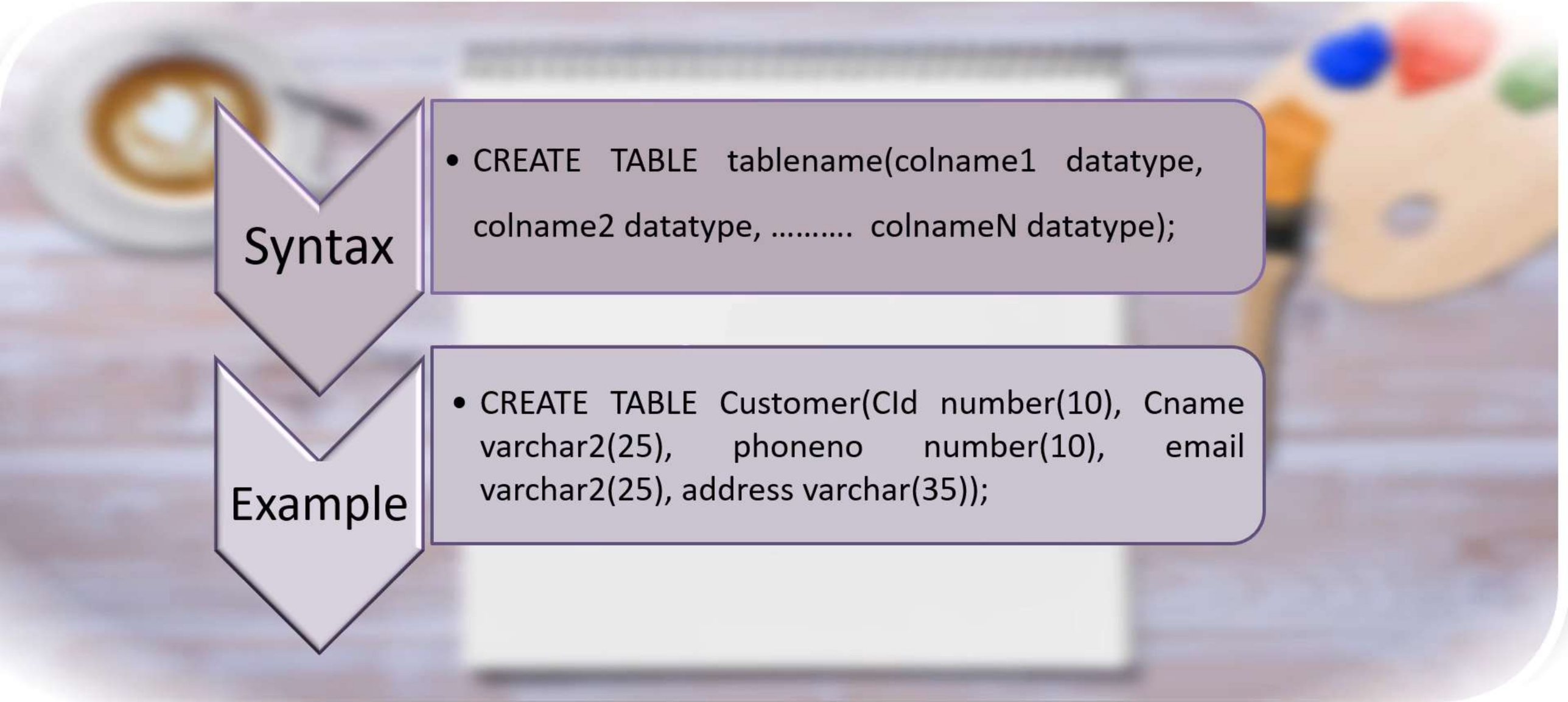
DATA TYPES

TEKNOTURF

DATATYPE	TYPE OF DATA	COLUMN DEFINITION	DESCRIPTION
Varchar2(size)	'John', 'Tom123' 'Alex@#'	NAME VARCHAR2(5)	When column 'NAME' stores data 'John' it occupies a size of 4 characters even though the column is created with a size of 5.
Char(size)	'John', 'Tom123' 'Alex@#'	NAME CHAR(5)	When column 'NAME' stores data 'John' it occupies the size of 5 characters even though the data has 4 characters.
Number(p)	123 45678	PHONENO NUMBER(8)	Column 'PHONENO' can store a number with maximum digit size of 8. Error will occur when the number of digit exceeds than specified size.
Number(p,s)	789.453 3.14	SALARY NUMBER(8,2)	Column Salary can store a value of 8 digits in which 2 digits should come after the decimal point.
Date	'17-Oct-20'	BIRTHDATE DATE	Date values are in the format 'DD -Mon-YY'. Column 'BIRTHDATE' will store date values.

CREATE A TABLE

TEKNOTURF



Syntax

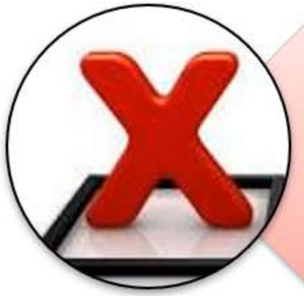
- CREATE TABLE tablename(colname1 datatype, colname2 datatype, ……. colnameN datatype);

Example

- CREATE TABLE Customer(CId number(10), Cname varchar2(25), phoneno number(10), email varchar2(25), address varchar(35));

NAMING CONVENTION

- Only letters, numbers and underscore are allowed in names
- Provides a meaningful table name.



- Asdkjad
- Employee details
- 123

- Employee_Details
- Employee
- Employee_V1

Record Insertion

TEKNOTURF

CID	CNAME	PHONENO	EMAIL	ADDRESS
1	Tom	9842012345	tom@gmail.com	Chennai
2	John	9876567890	john@gmail.com	Delhi
1	Ram	9123456780	ram@gmail.com	Pune

CONSTRAINT

TEKNOTURF

Constraints are the rules enforced on data columns on the table. These are used to limit the type of data that can go into a table. This ensures the accuracy and reliability of the data in the database.

Types of constraint

- Primary key
- Foreign key
- Unique
- Not null
- Check

A table can have one primary key, but it can have 'N' number of foreign keys, unique, not null and check constraints.

Two columns joined together can be made as single primary key.

PRIMARY KEY

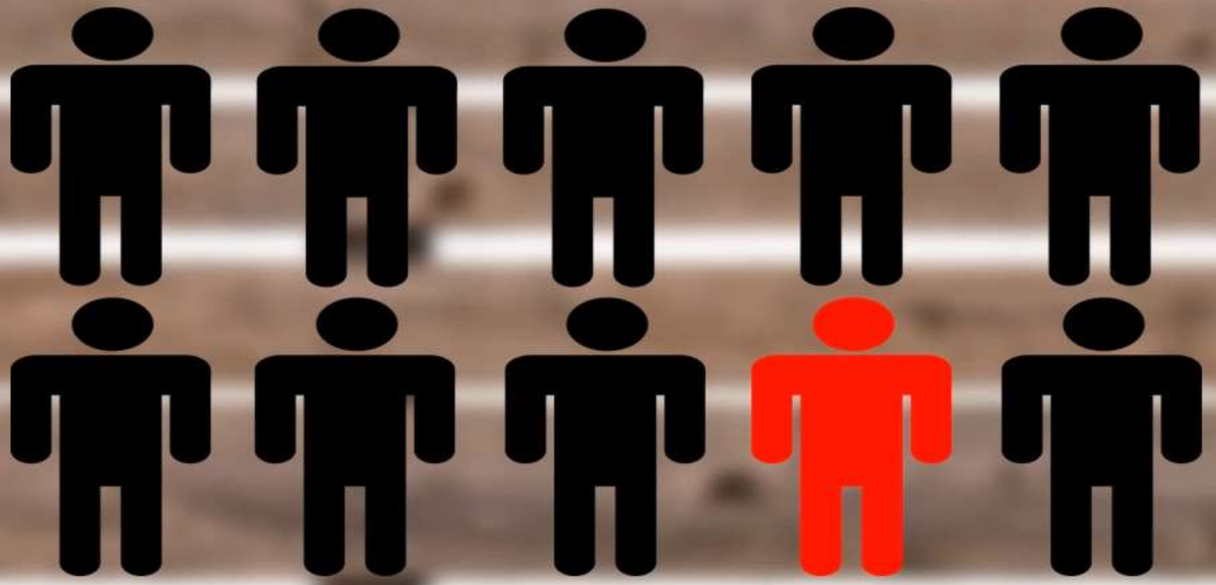
TEKNOTURF

CID	CNAME	PHONENO	EMAIL	ADDRESS
1	Tom	9876543210	tom@gmail.com	Chennai
2	John	8765798700	john@gmail.com	Delhi
1 ✖	Ram	7654323190	ram@gmail.com	Pune

Duplicate and NULL values are not allowed.

UNIQUE

CID	CNAME	PHONENO	EMAIL	ADDRESS
1	Tom	9876543210	tom@gmail.com	Chennai
2	John	8765798700	john@gmail.com	Delhi
1	Ram	7654323190	ram@gmail.com	Pune



Mark EMAIL attribute as unique if values in the attribute to be unique .

NOT NULL

TEKNOTURF

CID	CNAME	PHONENO	EMAIL	ADDRESS
1	Tom	9876543210	tom@gmail.com	Chennai
2	John	8765798700	john@gmail.com	Delhi
1	Ram	7654323190	ram@gmail.com	Pune

Mark CNAME attribute as Not Null to ensure the attribute should has value

CHECK

EnrollmentID	CId	PId	DueDate	PaidDate	Amount	Penalty
101	3	MBP	12-Dec-2017	11-Dec-2017	2000	0
102	1	PP	15-Mar-2018	13-Mar-2018	3000	0
103	2	PP	15-Feb-2018	22-Feb-2018	4000	200

Use check constraint to ensure, the attribute has only a positive value.

TEKNOTURF

FOREIGN KEY

TEKNOTURF

Customer

CID	CNAME	PHONENO	EMAIL	ADDRESS
1	Tom	9876543210	tom@gmail.com	Chennai
2	John	8765798700	john@gmail.com	Delhi

Policy

PID	PNAME	PPERIODINYEARS	MINAMOUNTPERMONTH
MBP	Money Back Plan	20	1000
PP	Personal Protect	15	1500

Policy Enrollment

ENROLLMENTID	CID	PID	DUEDATE	PAIDDATE	PENALTY
101	3	MBP	12-Dec-2017	11-Dec-2017	0
102	1		15-Mar-2018	13-Mar-2018	0

This problem can be resolved using foreign key.

There is no Customer with the id:3, then how can there be an entry in the Enrollment table?

FOREIGN KEY

- Foreign key is used to enforce the link between tables.
- The referenced table is called the ***parent table*** while the table with the foreign key is called the ***child table*.**
- The primary or unique column of the parent table can be created as the foreign key column in the child table.

ENROLLMENTID	CID	PID	DUEDATE	PAIDDATE	PENALTY
101	3 ✖	MBP	12-Dec-2017	11-Dec-2017	0
102	1	PP	15-Mar-2018	13-Mar-2018	0

We can mark ***CId*** and ***PId*** as foreign keys to ensure integrity.

PRIMARY AND FOREIGN KEY

TEKNOTURF

CID	CNAME	PHONENO	EMAIL	ADDRESS
1	Tom	9876543210	tom@gmail.com	Chennai
2	John	8765798700	john@gmail.com	Delhi

Foreign Key

ENROLLMENTID	CID	PID	DUEDATE	PAIDDATE	PENALTY
101	1	MBP	12-Dec-2017	11-Dec-2017	0
102	1	PP	15-Mar-2018	13-Mar-2018	0

Foreign Key

PID	PNAME	PPERIODINYEARS	MINAMOUNTPERMONTH
MBP	Money Back Plan	20	1000
PP	Personal Protect	15	1500

CREATE TABLE WITH CONSTRAINTS

TEKNOTURF

CREATE TABLE Customer(Cid number(10)PRIMARY KEY, CName varchar2(25)NOT NULL, phoneno number(10),email varchar2(25)UNIQUE);

CREATE TABLE Policy(Pid varchar2(10)PRIMARY KEY, Pname varchar2(25), PPeriodInYears varchar2(25) NOT NULL, MinAmount number(10)CHECK (MinAmount>0));

CREATE TABLE PolicyEnrollment(EnrollmentId number(5) PRIMARY KEY, Cid number(10) REFERENCES customer(Cid),Pid varchar2(10), Duedate date, Paiddate date, penalty number(10));

CONSTRAINT NAME

TEKNOTURF

- On creating a table with a constraint, oracle provides default name to it.
- The constraint names are stored in the built in table called "**USER_CONSTRAINTS**".
- User can override the default constraint name with the user defined constraint name.
- Example:

CREATE TABLE Policy(Pid varchar2(10) **CONSTRAINT pk_pid** primary key, Pname varchar2(25), PPeriodInYears varchar2(25) Not Null, MinAmount number(10) **CONSTRAINT ck_minamount** check (MinAmount>0));

ALTER

CREATE TABLE PolicyEnrollment(EnrollmentId number(5) primary key, Cid number(10) references customer(Cid),Pid varchar2(10), Duedate date, Paiddate date, penalty number(10));

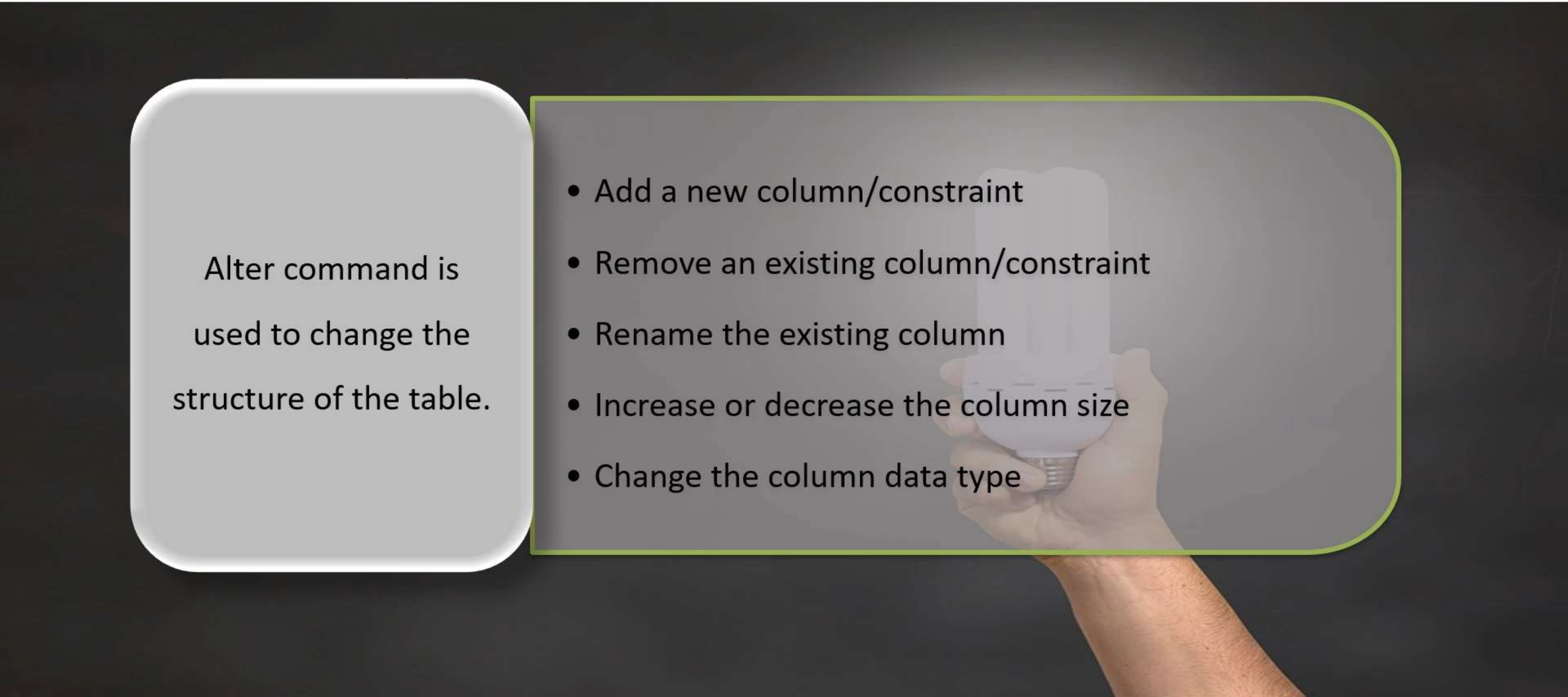
Alter table PolicyEnrollment add foreign key(Pid) references Policy(Pid);

ALTER

TEKNOTURF

Alter command is used to change the structure of the table.

- Add a new column/constraint
- Remove an existing column/constraint
- Rename the existing column
- Increase or decrease the column size
- Change the column data type



ALTER - EXAMPLE

TEKNOTURF

To add a column age to customer table.

```
ALTER TABLE  customer ADD  age  number(2);
```

To increase the column size of email to 30.

```
ALTER TABLE customer MODIFY email varchar2(30);
```

To change the column name from email to email id.

```
ALTER TABLE customer RENAME COLUMN email TO emailid;
```

TABLE LEVEL CONSTRAINT

How can I check whether the paid date is greater than the due date in the Policy Enrollment Table?

You can use table level constraint to check this.

```
CREATE TABLE PolicyEnrollment(EnrollmentId number(5) primary key, bid varchar2(10) references book(bid), mid number(10) references member(mid), issuedate date, returndate date, penalty number(10), check(returndate>issuedate));
```

Constraints can be column level or table level. Column level constraints are applied only to one column, whereas table level constraints are applied to the whole table.

TRUNCATE

Removes all rows from the table

Restriction

- You cannot truncate the table if it is linked with another table

Syntax

- TRUNCATE TABLE <Table_name>;

Example

- TRUNCATE TABLE Customer;

DROP

Drops the entire table structure

Syntax

- DROP TABLE <TABLE_NAME>

Example

- DROP TABLE Customer;

TRUNCATE VS DROP

TEKNOTURF

Table

Table after truncate

Table after drop

In truncate only the data is removed,
whereas,
in drop the entire structure is removed.

Summary

- Data types
- Create Table
- Alter Table
- Drop and Truncate Table
- Various constraints

TEKNOTURF

DATA MANIPULATION LANGUAGE

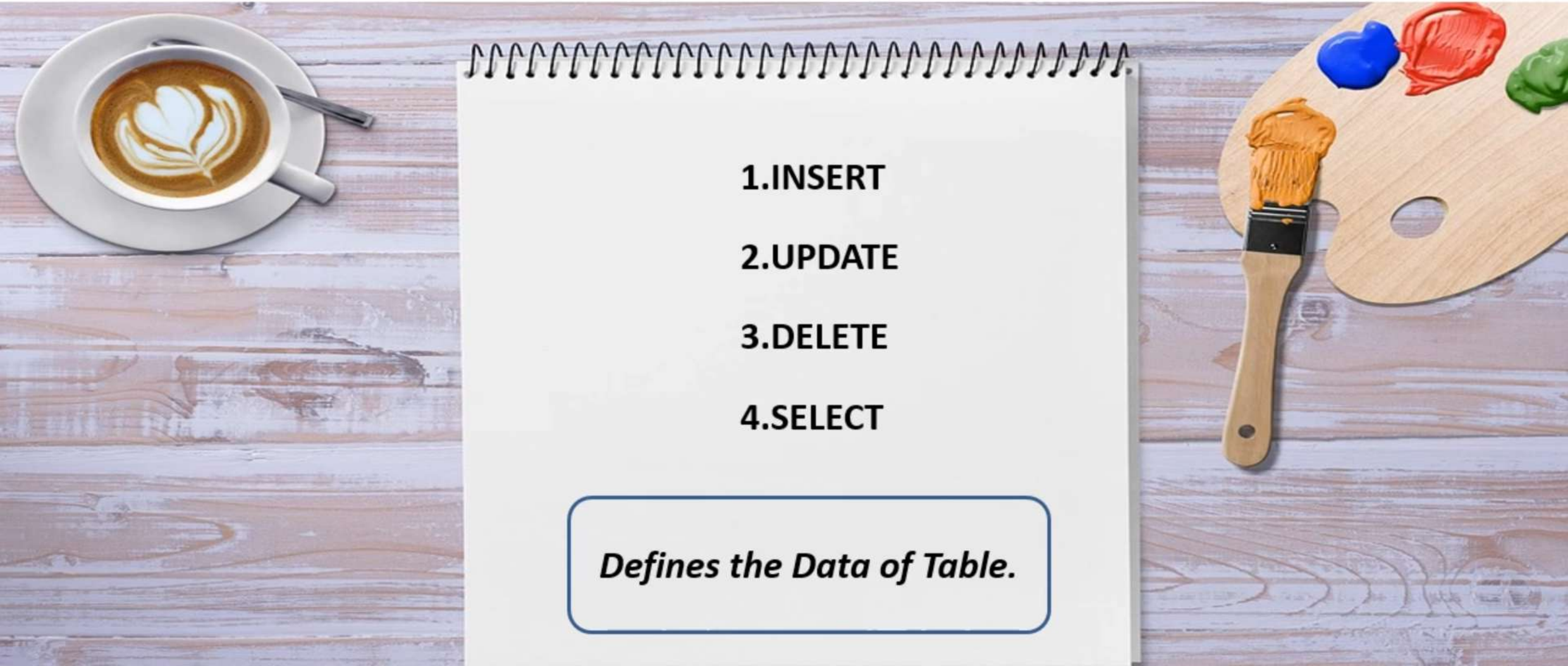
TEKNOTURF

IN THIS MODULE YOU WILL LEARN

- Data Manipulation Language (DML)
- Add a New Row to a Table
- Update Rows in a Table
- Remove a Row from a Table
- The MERGE Statement
- Database Transactions
- Locking

DATA MANIPULATION LANGUAGE

TEKNOTURF



1.INSERT

2.UPDATE

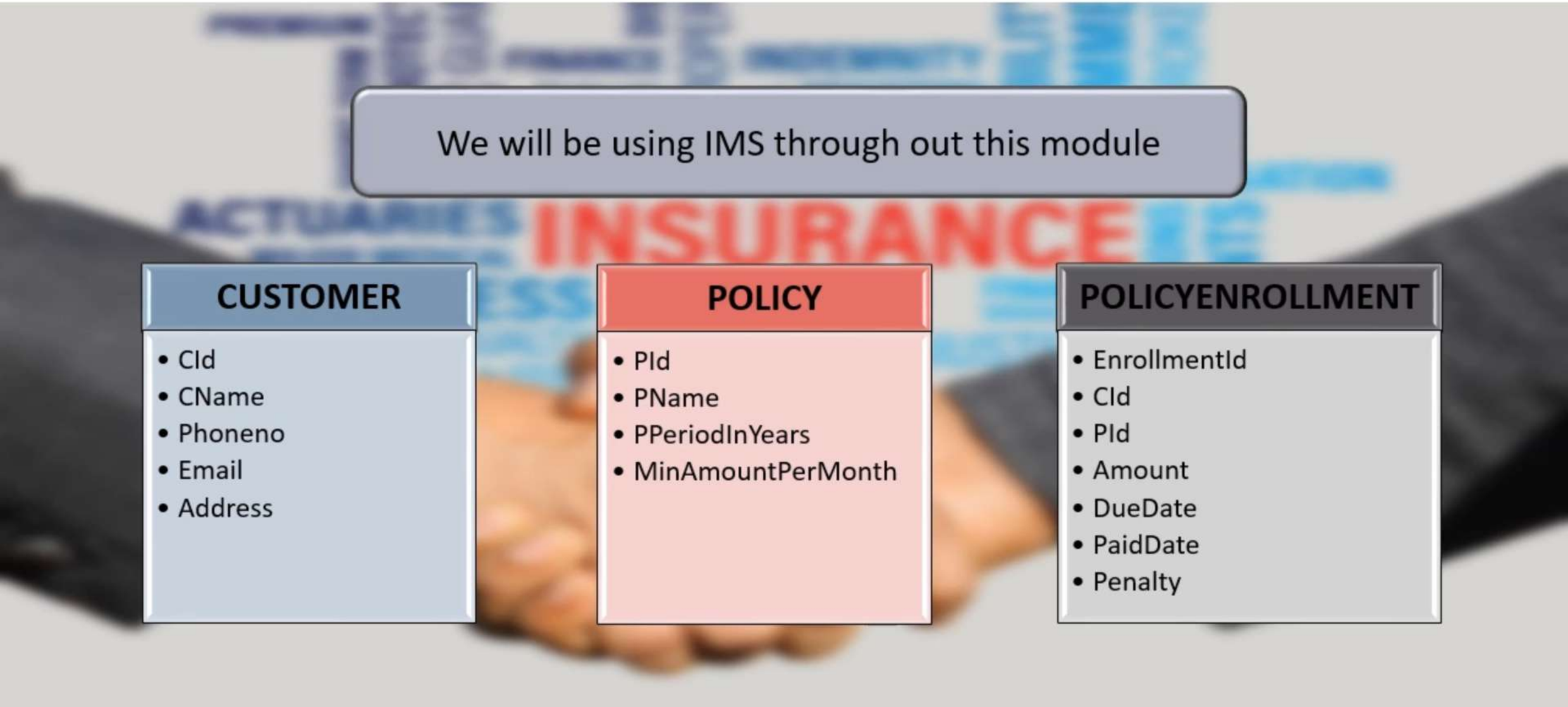
3.DELETE

4.SELECT

Defines the Data of Table.

INSURANCE MANAGEMENT SYSTEM(IMS)

TEKNOTURF



We will be using IMS through out this module

CUSTOMER
• CId • CName • Phoneno • Email • Address

POLICY
• PId • PName • PPeriodInYears • MinAmountPerMonth

POLICYENROLLMENT
• EnrollmentId • CId • PId • Amount • DueDate • PaidDate • Penalty

SAMPLE RECORDS

TEKNOTURF

CID	CNAME	PHONENO	DOB	EMAILID	ADDRESS
1	Tom	9876523190	17-MAR-87	tom@gmail.com	chennai
2	John	8765432190	26-JAN-86	john@yahoo.com	delhi
3	Ram	7654321890	14-DEC-85	ram@gmail.com	pune
4	Tiny	9012365478	28-MAY-86	NULL	chennai

Customer

PID	PNAME	PPERIODINYEARS	MINAMOUNTPERMONTH
MBP	Money Back Plan	20	1000
PP	Personal Protect	15	1500

Policy

ENROLLMENTID	CID	PID	DUEDATE	PAIDDATE	AMOUNT	PENALTY
101	3	MBP	12-Dec-2017	11-Dec-2017	2000	0
102	1	PP	15-Mar-2018	13-Mar-2018	3000	0
103	2	PP	15-Feb-2018	22-Feb-2018	4000	200

Policy Enrollment

SAMPLE RECORDS

TEKNOTURF

CID	CNAME	PHONENO	DOB	EMAILID	ADDRESS
1	Tom	9876523190	17-MAR-87	tom@gmail.com	chennai
2	John	8765432190	26-JAN-86	john@yahoo.com	delhi
3	Ram	7654321890	14-DEC-85	ram@gmail.com	pune
4	Tiny	9012365478	28-MAY-86	NULL	chennai

Customer

PID	PNAME	PPERIODINYEARS	MINAMOUNTPERMONTH
MBP	Money Back Plan	20	1000
PP	Personal Protect	15	1500

Policy

ENROLLMENTID	CID	PID	DUEDATE	PAIDDATE	AMOUNT	PENALTY
101	3	MBP	12-Dec-2017	11-Dec-2017	2000	0
102	1	PP	15-Mar-2018	13-Mar-2018	3000	0
103	2	PP	15-Feb-2018	22-Feb-2018	4000	200

Policy Enrollment

INSERT ROWS TO TABLE

Syntax:

INSERT INTO *TableName [(column 1, column2...)]* **VALUES** *(value1 , value2...)*;

Example:

INSERT INTO Customer **VALUES** (1,'Tom',9876523190,'Tom@gmail.com','Mumbai');

INSERT ROWS TO TABLE

TEKNOTURF

Data for DATE, CHAR and VARCHAR types should be always enclosed within single quotes.

Specific insertion.

```
INSERT INTO Customer (Id,Cname,phoneno,address)
VALUES (2,'Mini',9128374605,’Chennai');
```

UPDATE ROWS IN A TABLE

TEKNOTURF

UPDATE *TableName* SET *ColumnName = value [,ColumnName = value, ...]* [WHERE *condition*];

Modify existing rows with the UPDATE statement.

- UPDATE Customer SET emailid='Tiny15@gmail.com' WHERE cname='Tiny';

All rows in the table are modified if we omit the WHERE clause.

- UPDATE Customer SET emailid='Tiny15@gmail.com';

DELETE ROWS FROM TABLE

TEKNOTURF

DELETE [FROM] ***table*** **[WHERE** ***condition];***

Remove existing rows from a table by using the DELETE statement.	• DELETE FROM Customer WHERE CId=2;
To delete all rows from the table.	• DELETE FROM Customer;

Deletion is not possible if the row to be deleted is referred in the child table.

Deletion of the parent record is made possible by using a foreign key reference option.

DELETE ROWS FROM TABLE

TEKNOTURF

REFERENCE OPTION

- **CASCADE** : Deletes the row from the parent table, and automatically deletes the matching rows in the child table.
- **SET NULL** : Deletes the row from the parent table, and sets the foreign key column in the child table to NULL.
- **RESTRICT** : Rejects the delete or update operation for the parent table. This is default option.

SYNTAX

- REFERENCES tbl_name (index_col_name,...) [ON DELETE reference_option]
- reference_option: RESTRICT | CASCADE | SET NULL

ON DELETE CASCADE

TEKNOTURF

When a record is removed from the customer table, it should also be removed from the related records in the enrollment table.

Use "**ON DELETE CASCADE**" while creating a table.

```
CREATE TABLE PolicyEnrollment(EnrollmentId number(5) primary key, Cid
number(10) references customer(Cid) ON DELETE CASCADE, Pid varchar2(10),
Duedate date, Paiddate date, penalty number(10));
```

ON DELETE SET NULL

TEKNOTURF

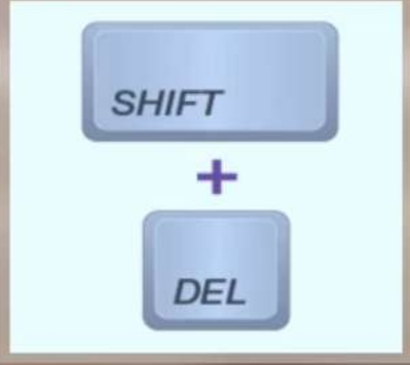
If you want to set null values instead of removing the record from the transaction table, how do you do that?

Use "**ON DELETE SET NULL**" while creating the table.

```
CREATE TABLE PolicyEnrollment(EnrollmentId number(5) primary key, Cid
number(10) references customer(Cid) ON DELETE SET NULL, Pid varchar2(10),
Duedate date, Paiddate date, penalty number(10));
```

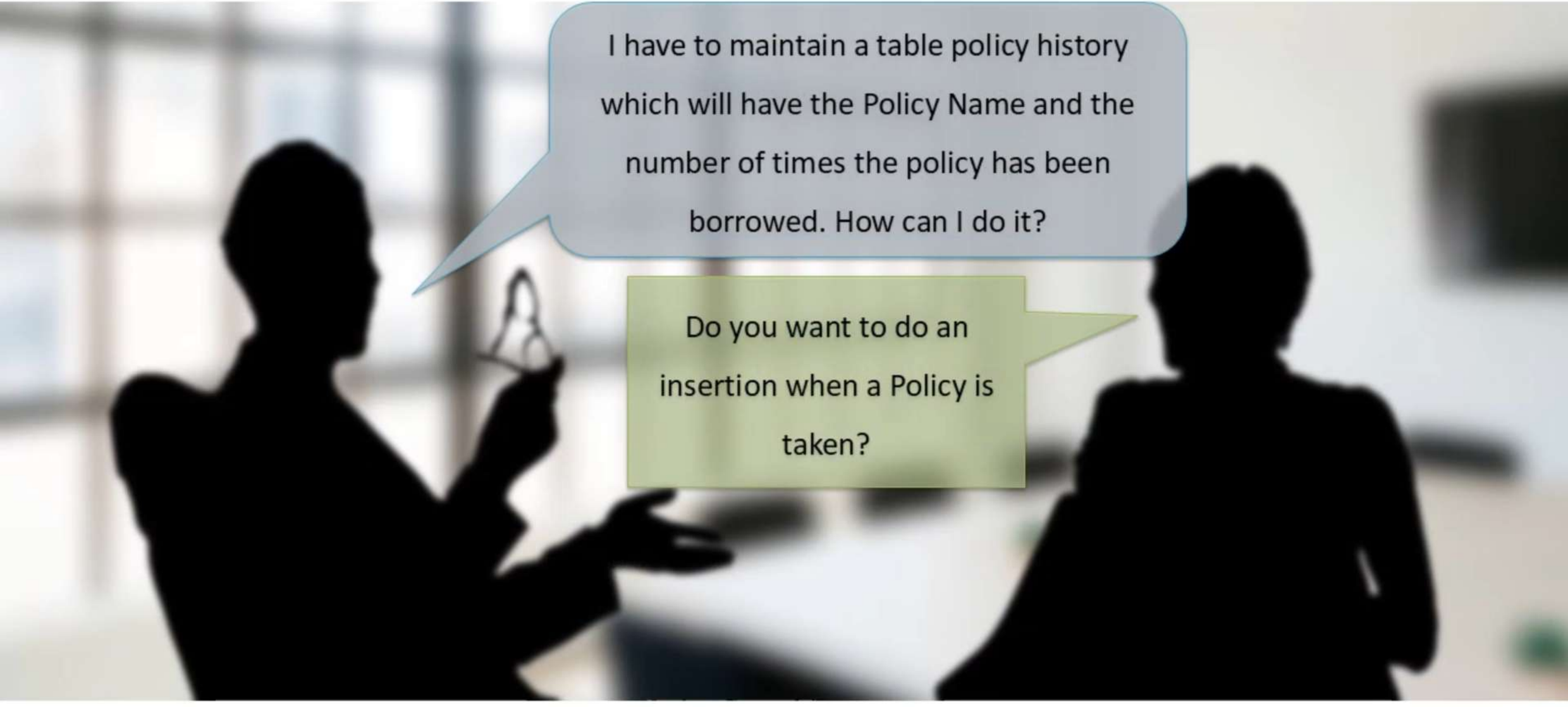
DELETE VS TRUNCATE

TEKNOTURF

DELETE	Deletes all rows or a specific row from the table.
	Can be reverted to its previous state.
	Rows that are referred in the child table cannot be removed.
TRUNCATE	Removes all rows from the table.
	Cannot be reverted.
	Will not work if the truncated table is referenced.

MERGE

TEKNOTURF



MERGE

TEKNOTURF

MERGE

TEKNOTURF

Provides the ability to conditionally update or insert data into a database table.

Performs an UPDATE if the row exists and an INSERT if it is a new row.

- Avoids separate updates.
- Increases performance and ease of use.

The MERGE statement is deterministic .

- Cannot update the same row of the target table multiple times in the same MERGE Statement.

MERGE

TEKNOTURF

Syntax

```
MERGE INTO table_name  table_alias
        USING (table|view|sub_query)  alias
        ON (join condition)
WHEN MATCHED THEN
        UPDATE SET
                col1 = col_val1,
                col2 = col2_val
WHEN NOT MATCHED THEN
        INSERT (column_list)
                VALUES (column_values);
```

TEKNOTURF

MERGE

Example

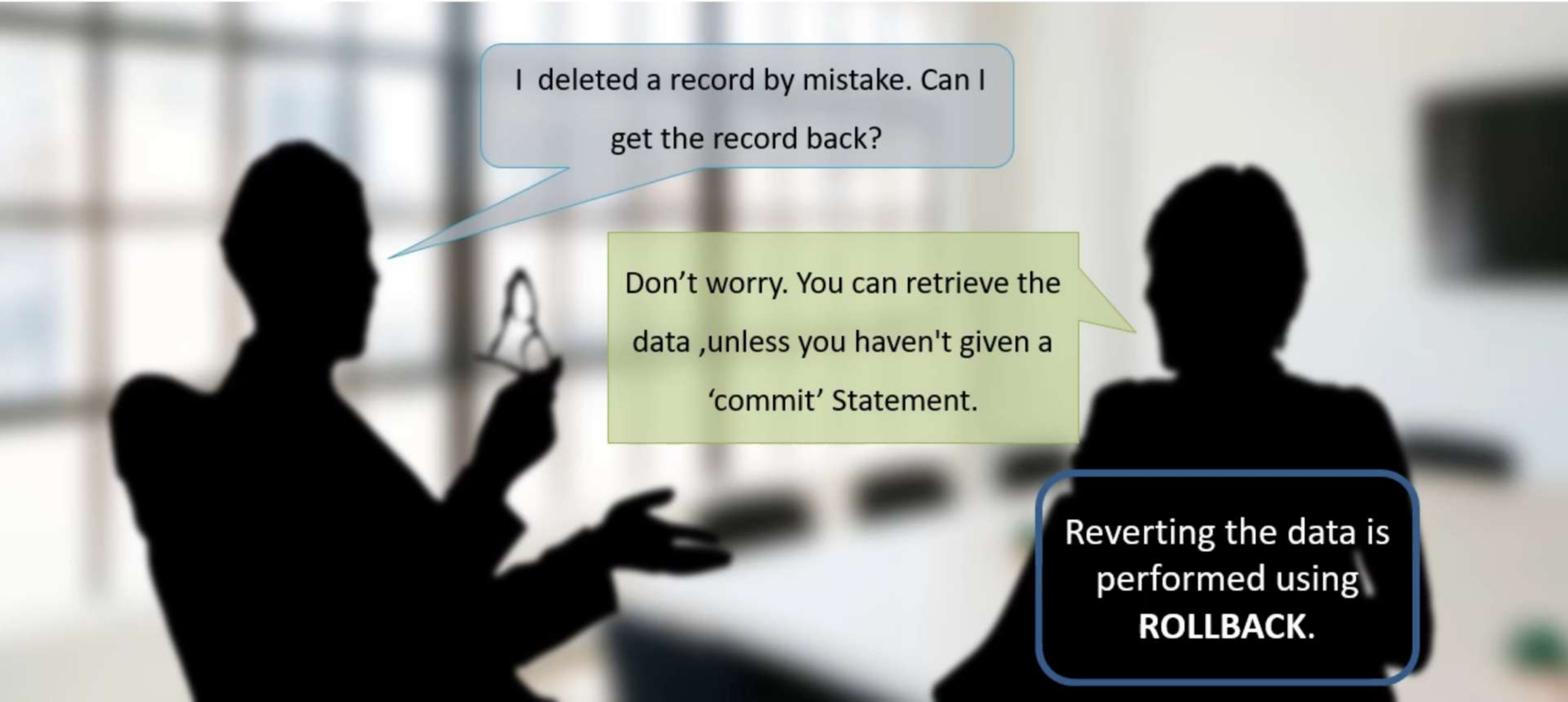
```
MERGE INTO policyhistory as ph
        USING Policy p
        ON (ph.Pname = p.Pname)
WHEN MATCHED THEN
        UPDATE SET ph.count=ph.count+1
WHEN NOT MATCHED THEN
        INSERT(ph.pname,ph.count)  VALUES(p.pname,1);
```

DATABASE TRANSACTION

TEKNOTURF

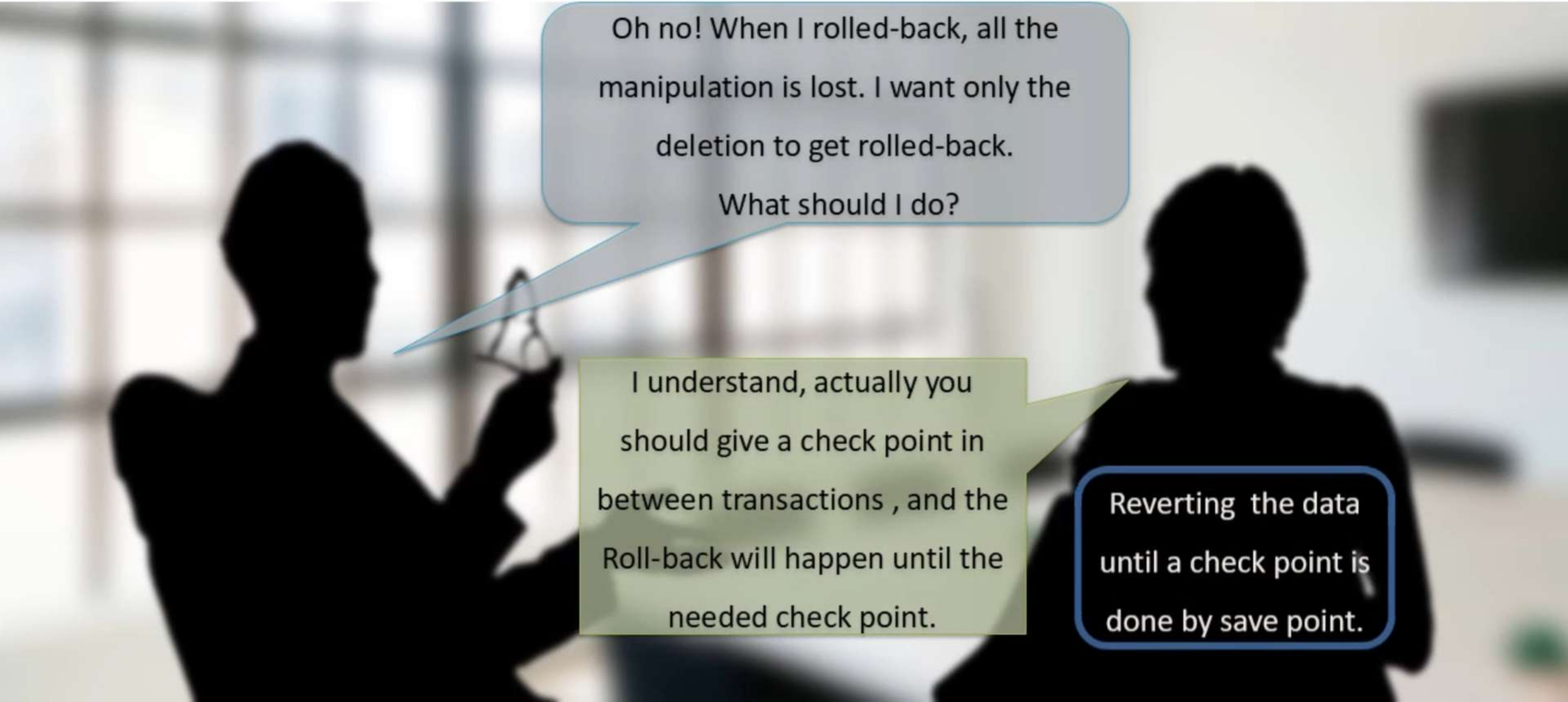
DATABASE TRANSACTION

TEKNOTURF



DATABASE TRANSACTION

TEKNOTURF



DATABASE TRANSACTION

TEKNOTURF

- A transaction is a logical unit of work.
- Oracle ensures data consistency based on transactions.
- Transaction does consistent data change.
- Transaction begins when DML statement is executed.
- Transaction should end with either **Commit or Rollback.**
- DDL commands are by default auto commit.

DATABASE TRANSACTION

TEKNOTURF

COMMIT	• Ends the current transaction by making all pending data changes permanent.
SAVEPOINT *Savepoint_name*	• Marks a save point within the current transaction.
ROLLBACK	• ROLLBACK ends the current transaction by discarding all pending data changes.
ROLLBACK TO *Savepoint_name*	• ROLLBACK TO SAVEPOINT rolls back the current transaction to the specified savepoint, thereby discarding any changes or savepoints created after the savepoint to which you are rolling back.

LOCKING

TEKNOTURF

Prevent destructive interaction between concurrent transactions when accessing shared resources.

Locking is performed automatically and requires no user action.

Uses the lowest level of restrictiveness.

Locks are held for a duration of a transaction.

Types

- Explicit locking
- Implicit locking

TEKNOTURF

SUMMARY

- Data Manipulation Language (DML)
- Add a New Row to a Table
- Update Rows in a Table
- Remove a Row from a Table
- The MERGE Statement
- Database Transactions
- Locking

Description Edit Grading view

Insert Records - Department

Requested files: insert.sql
(Download)

Type of work: Individual work
Contains the run and evaluate script of Oracle DML. Can be used as example

Insert the following records to the department table

Department_id	Department_name	Department_block_number
1	Computer Science	3
2	Information Technology	3
3	Software Engineering	3

Refer the below schema diagram:

Description Edit Grading view

File List Save Compile & Run Evaluate Full screen Descriptio

insert.sql

```
1  insert into DeParTmenT values (1, 'Computer Science',3);
2  insert into department values (2, 'Information Technology',3);
3  insert into department values (3, 'Software engineering',3);
```

Question Description

Insert the following records to the department table

Department_id	Department_name	Department_blo
1	Computer Science	3
2	Information Technology	3
3	Software Engineering	3

Description Edit Grading view

File List Save

insert.sql

```
1 insert into Dep
2 insert into dep
3 insert into dep
```

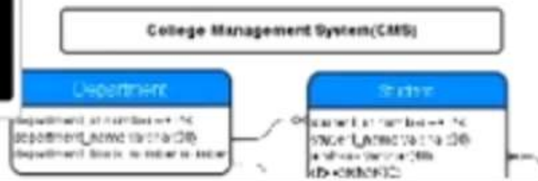
>_ Console: connection closed (Running)

```
Row(s) inserted.
```

Question Description

ert the following records to the department
le

partment_id	Department_name	Department_blo
	Computer Science	3
	Information Technology	3
	Software Engineering	3



04:43

Description Edit Grading view

File List | Save | Compile & Run | Evaluate | Full screen | Description

insert.sql

```
1  insert into DeParTmenT values (1, 'Computer Science',3);
2  insert into department values (2, 'Information Technology',3);
3  insert into department values (3, 'Software engineering',3);
```

Proposed grade: 0 / 100

Result Description

--- Expected output ---

```
Displaying the Updated rows

DEPARTMENT_ID DEPARTMENT_NAME
------------- ----------------------
            1 Computer Science
            2 Information Technology
            3 Software Engineering
```

---Difference in Output---

```
Displaying the Updated rows

DEPARTMENT_ID DEPARTMENT_NAME
------------- ----------------------
            1 Computer Science
            2 Information Technology
            3 Software eEngineering
```

Description Edit Grading view

File List | Save | Compile & Run | Evaluate | Full screen | Description

insert.sql

```
1  insert into DeParTmenT values (1, 'Computer Science',3);
2  insert into department values (2, 'Information Technology',3);
3  insert into department values (3, 'Software Engineering',3);
```

Proposed grade: 100 / 100

Result Description

Summary of tests

```
+-----------------------------+
| 1 tests run / 1 test passed |
+-----------------------------+
```

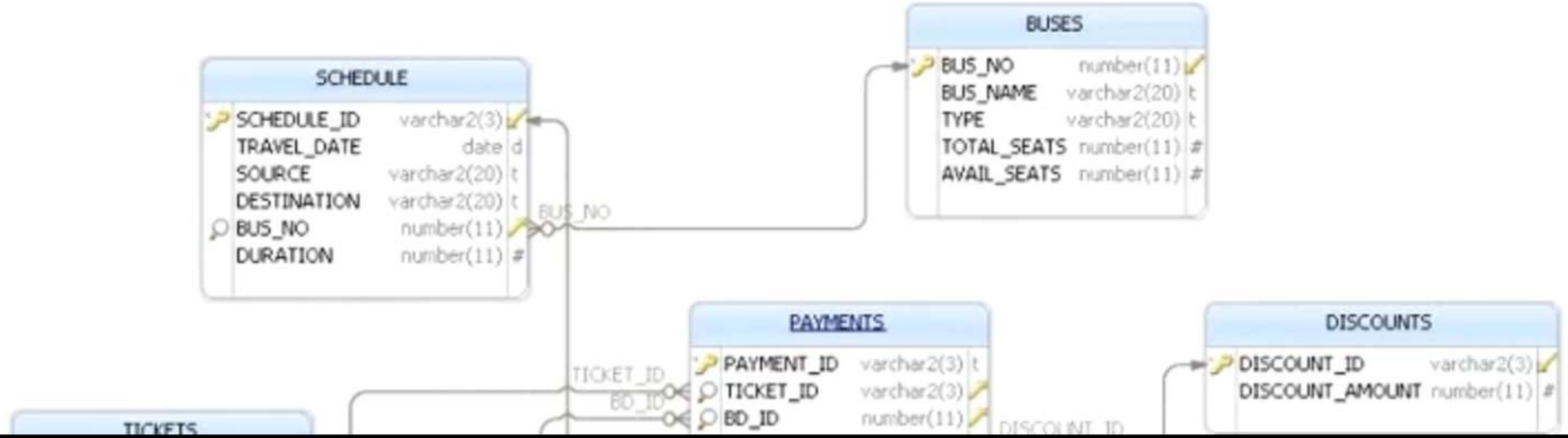
Description Edit Grading view

Update Buses table

Requested files: update1.sql
(Download)

Type of work: Individual work
Contains the run and evaluate script of Oracle DML. Can be used as example

Write a query to change the bus type to 'AC' for all buses.



Description Edit Grading view

File List Save Compile & Run Evaluate Full screen

Save (Ctrl-S)

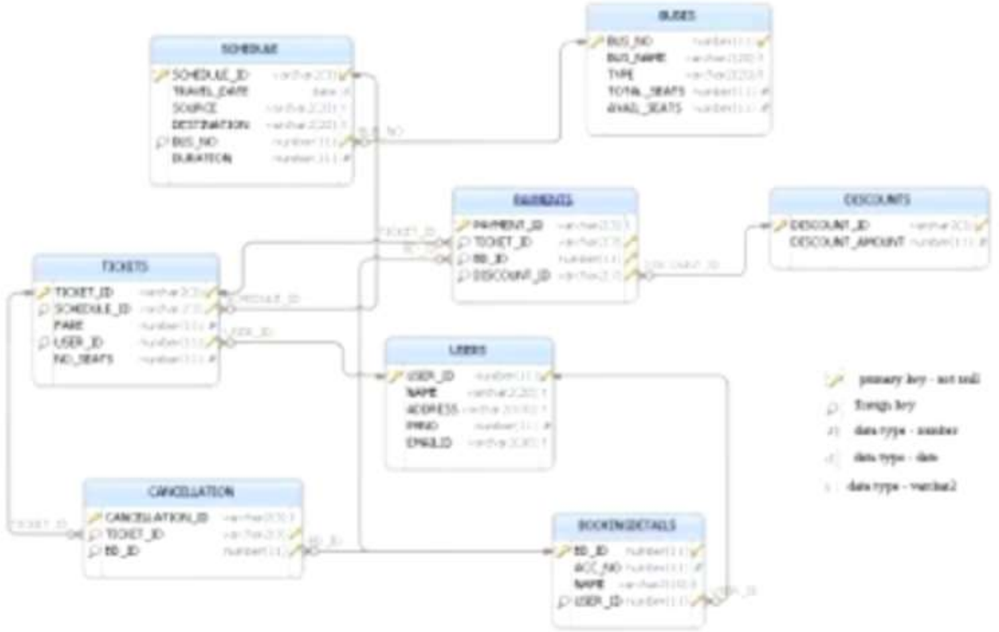
update1.sql

```
1 update buses set type='AC';
```

forms/edit.php?id=13225&useri...

Question Description

Write a query to change the bus type to 'AC' for all buses.



Description Edit Grading view

File List Save

update1.sql

```
1 update buses se
```

>_ Console: connection closed (Running)

```
Row(s) updated.
```

Description Edit Grading view

File List | Save | Compile & Run | Evaluate | Full screen | Description

update1.sql

```
1 update buses set type='AC';
```

Proposed grade: 100 / 100

Result Description

Summary of tests

```
+-----------------------------+
| 1 tests run / 1 test passed |
+-----------------------------+
```

TEKNOTURF

SQL

SELECT STATEMENT

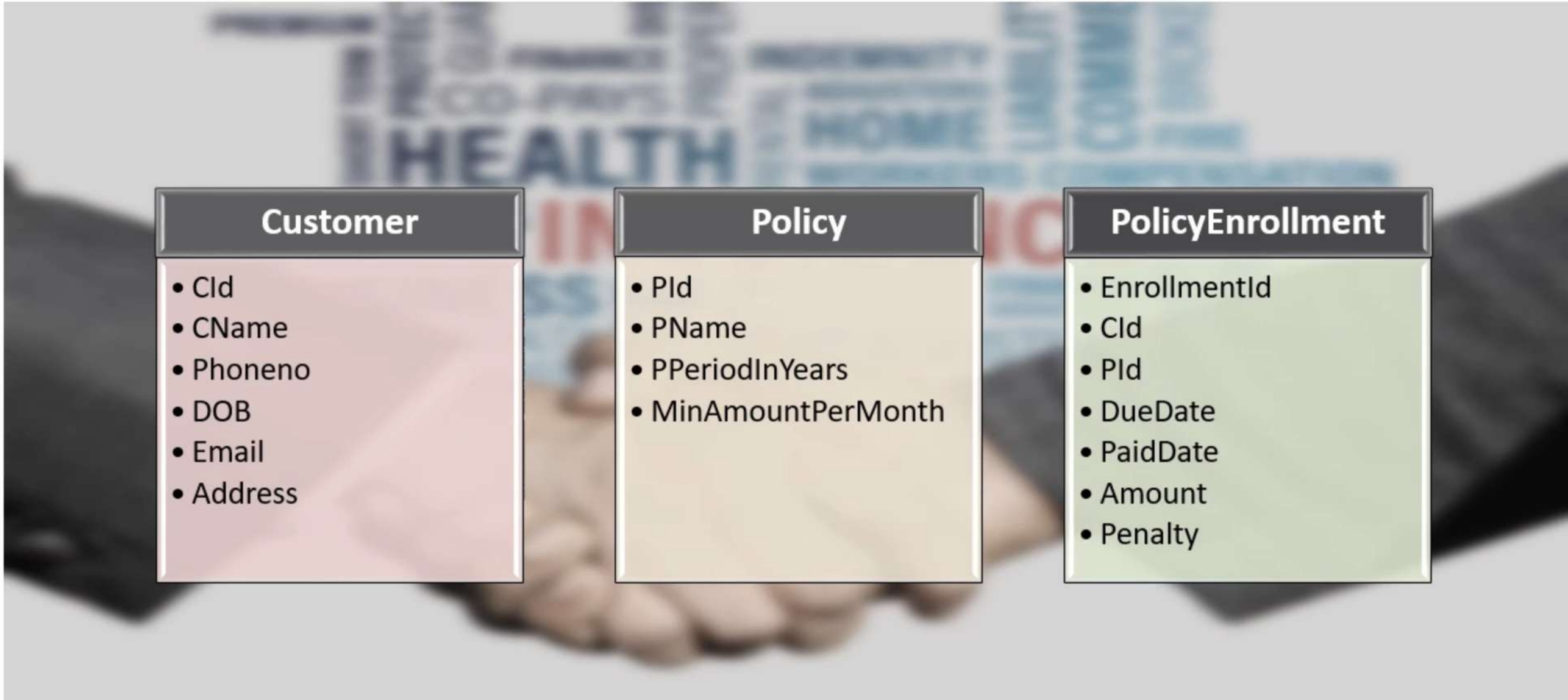
Objective

TEKNOTURF

- Select Statement
- Where Clause
- Types of Operators
- Order by Clause
- Distinct

Insurance Management System(IMS)

TEKNOTURF



Customer
• CId • CName • Phoneno • DOB • Email • Address

Policy
• PId • PName • PPeriodInYears • MinAmountPerMonth

PolicyEnrollment
• EnrollmentId • CId • PId • DueDate • PaidDate • Amount • Penalty

Sample Records

CID	CNAME	PHONENO	DOB	EMAILID	ADDRESS
1	Tom	9876523190	17-MAR-87	tom@gmail.com	chennai
2	John	8765432190	26-JAN-86	john@yahoo.com	delhi
3	Ram	7654321890	14-DEC-85	ram@gmail.com	pune
4	Tiny	9012365478	28-MAY-86	NULL	chennai

Customer

PID	PNAME	PPERIODINYEARS	MINAMOUNTPERMONTH
MBP	Money Back Plan	20	1000
PP	Personal Protect	15	1500

Policy

ENROLLMENTID	CID	PID	DUEDATE	PAIDDATE	AMOUNT	PENALTY
101	3	MBP	12-Dec-2017	11-Dec-2017	2000	0
102	1	PP	15-Mar-2018	13-Mar-2018	3000	0
103	2	PP	15-Feb-2018	22-Feb-2018	4000	200

Policy Enrollment

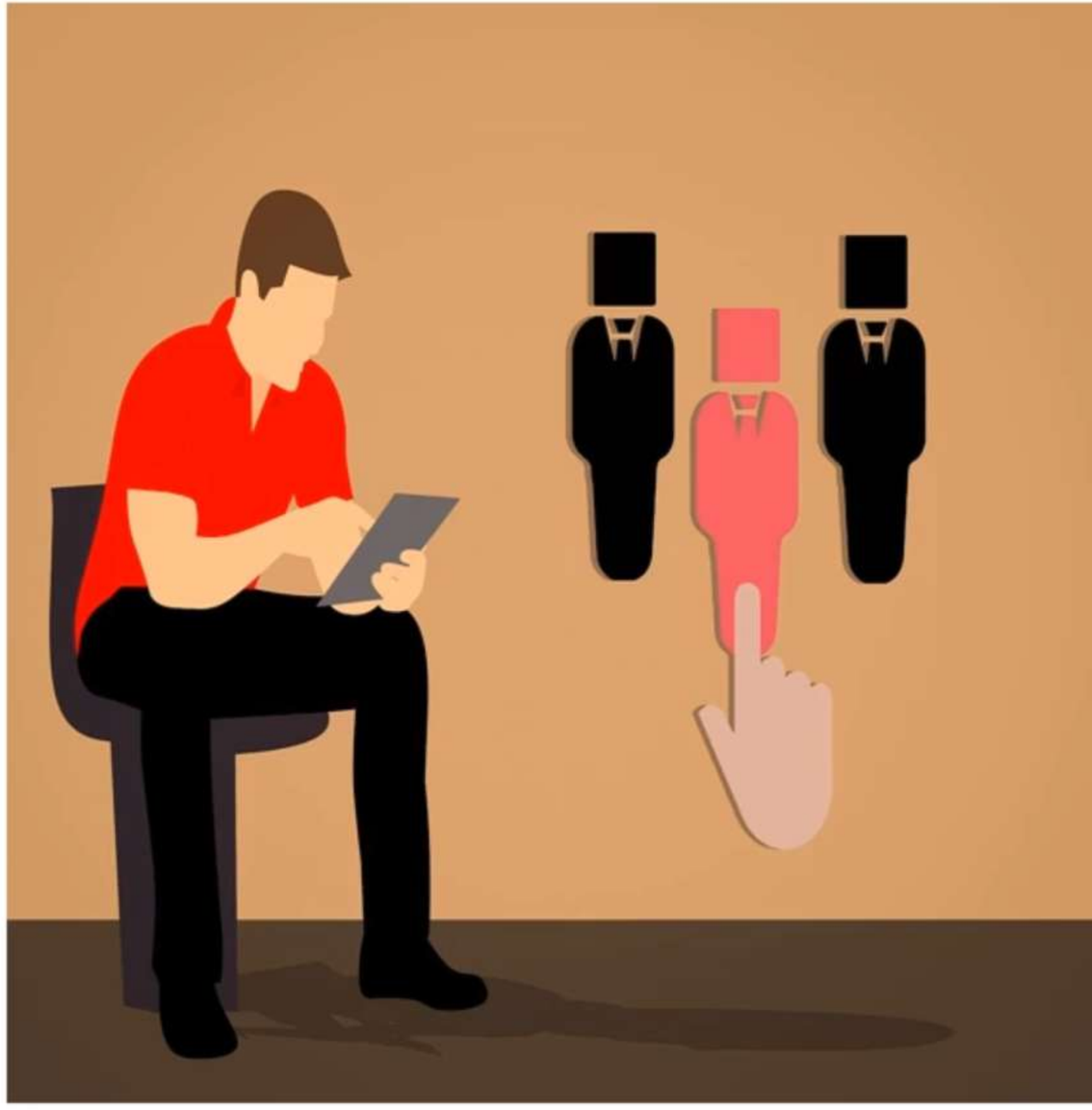
SELECT Statement

TEKNOTURF

- The **SELECT** statement of SQL is used to retrieve data from one or more tables or views.
- The result returned is stored in a temporary result table, called the result-set.

SELECT can be used for retrieving:

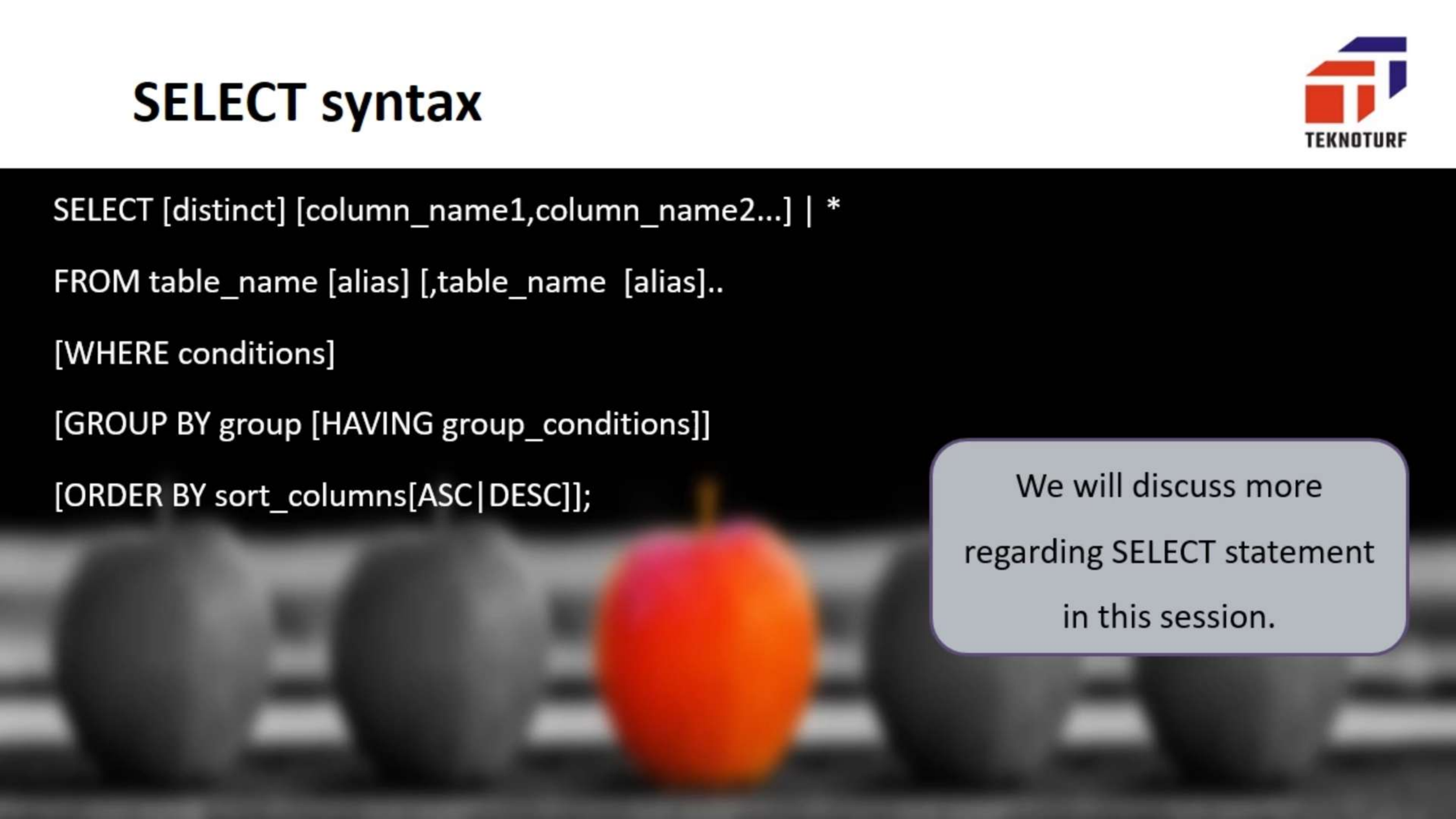
- all rows from a table.
- specific rows from a table.
- specific values from a table.



SELECT syntax

TEKNOTURF

```
SELECT [distinct] [column_name1,column_name2...] | *
FROM table_name [alias] [,table_name  [alias]..
[WHERE conditions]
[GROUP BY group [HAVING group_conditions]]
[ORDER BY sort_columns[ASC|DESC]];
```



We will discuss more regarding SELECT statement in this session.

SELECT

SQL statements are not case sensitive, but the data stored in the database is case sensitive. For example: a member with the name **tom** and **TOM** are different, but SELECT and select are the same statement.

Retrieve all the records

TEKNOTURF

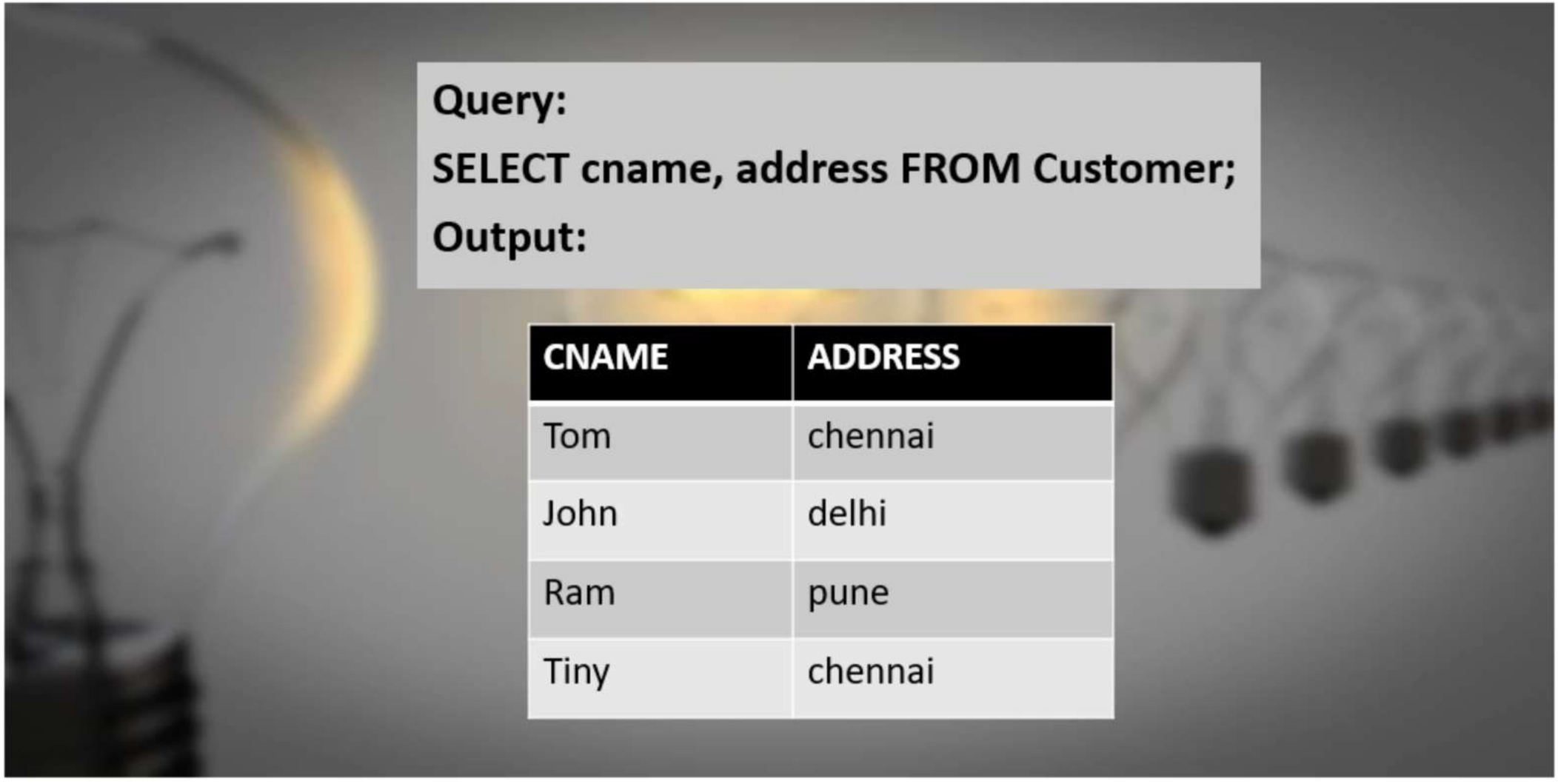
Query:

```
SELECT * FROM Customer;
```

Output:

CID	CNAME	PHONENO	DOB	EMAILID	ADDRESS
1	Tom	9876523190	17-MAR-87	tom@gmail.com	chennai
2	John	8765432190	26-JAN-86	john@yahoo.com	delhi
3	Ram	7654321890	14-DEC-85	ram@gmail.com	pune
4	Tiny	9012365478	28-MAY-86	NULL	chennai

Retrieve Specific Columns



Query:

SELECT cname, address FROM Customer;

Output:

CNAME	ADDRESS
Tom	chennai
John	delhi
Ram	pune
Tiny	chennai

Arithmetic Expressions

TEKNOTURF

I want to know the amount of the policy including it's GST of 10%.

You can perform this operation using Arithmetic Operator in SELECT clause.

PID	PNAME	PPERIODINYEARS	MINAMOUNTPERMONTH
MBP	Money Back Plan	20	1000
PP	Personal Protect	15	1500

```
SELECT pid, Minamountpermonth+(Minamountpermonth*0.1) FROM policy;
```

PID	MINAMOUNTPERMONTH +(MINAMOUNTPERMONTH*0.1)
MBP	1100
PP	1650

Column Alias

TEKNOTURF

Oh! Why is the column name meaningless in the previous query?

Provide an alias name for the column.

```
SELECT pid, Minamountpermonth+(Minamountpermonth*0.1)  PolicyAmount FROM policy;
```

PID	POLICYAMOUNT
MBP	1100
PP	1650

Arithmetic expressions containing a Null Value evaluates to Null.

Enclose the alias name in double quotes, if it contains space between words.

Concatenation Operator

TEKNOTURF

How to display the customer name along with their address in the following format "CustomerName" lives in "Address"?

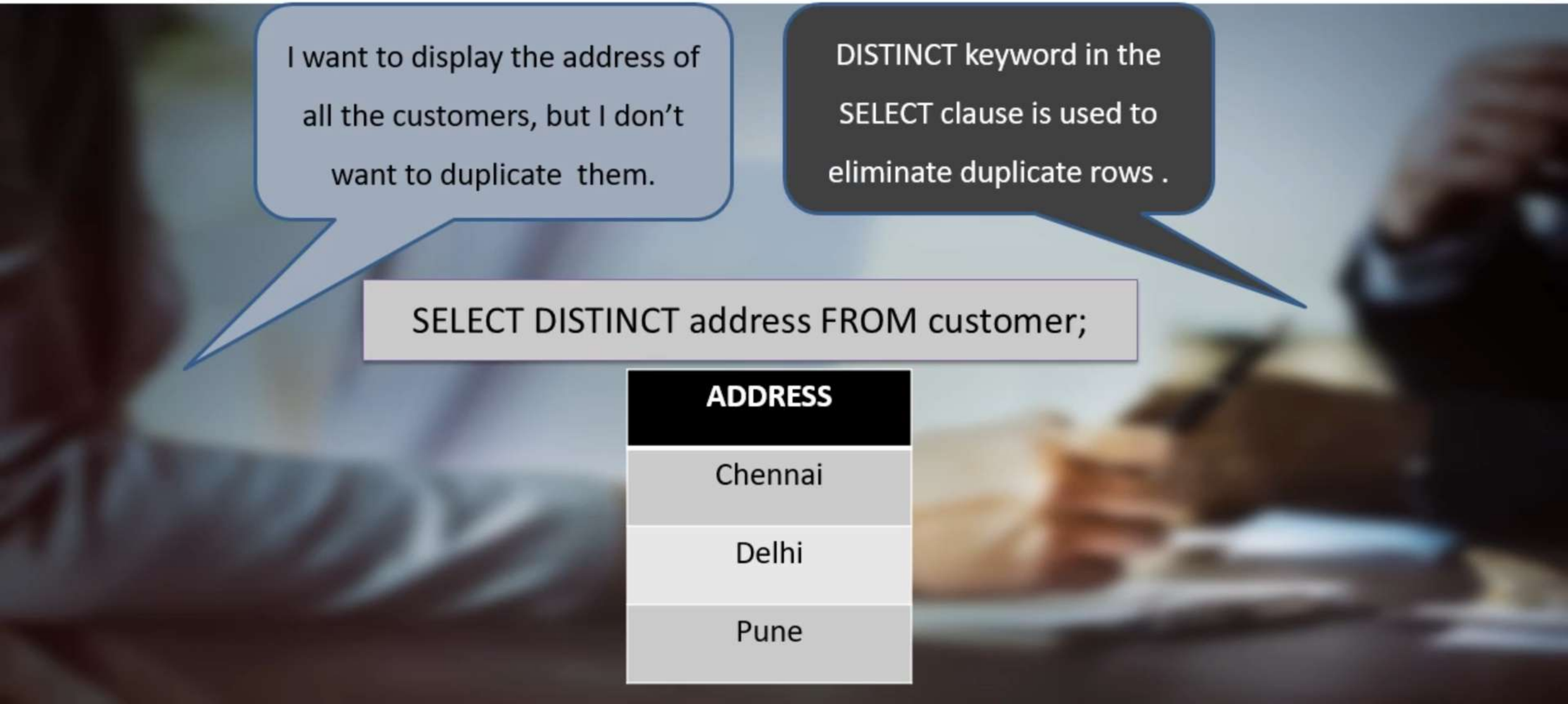
|| operator is used to concatenate columns or character strings.

```
SELECT Cname||' lives  in'||Address as  CustomerAddress FROM customer;
```

CUSTOMERADDRESS
Tom lives in Chennai
John lives in Delhi
Ram lives in Pune
Tiny lives in Chennai

Eliminating Duplicate Rows

TEKNOTURF



I want to display the address of all the customers, but I don't want to duplicate them.

DISTINCT keyword in the SELECT clause is used to eliminate duplicate rows .

```
SELECT DISTINCT address FROM customer;
```

ADDRESS
Chennai
Delhi
Pune

WHERE CLAUSE

Restrict the rows returned by using the WHERE clause. The **'WHERE'** clause follows the **'FROM'** clause.

SELECT *|{[**DISTINCT**] column|expression [alias],...}

FROM table [**WHERE** condition(s)];

Value in the condition is compared using **Relational operators.**

Logical operators are used to specify more than one condition in WHERE clause.

Where CLAUSE

TEKNOTURF

What is the query to retrieve customers who are only from Chennai?

Use 'WHERE clause', which is used to extract only those records that fulfill a specific condition.

SELECT cname, emailid, address FROM customer WHERE address='Chennai';

CNAME	EMAILID	ADDRESS
Tom	tom@gmail.com	Chennai
Tiny	tiny@yahoo.co.in	Chennai

Comparison Operators

TEKNOTURF

Operator	Explanation
=	Equal to.
>	Greater than.
>=	Greater than or equal to.
<	Less than.
<=	Less than or equal to.
!=, <>	Not equal to.
Between … and	Range of values.
In(set)	Match any of the list values.
Like	Match a character pattern.
Isnull	Is a null value.

Example - Less than operator

TEKNOTURF

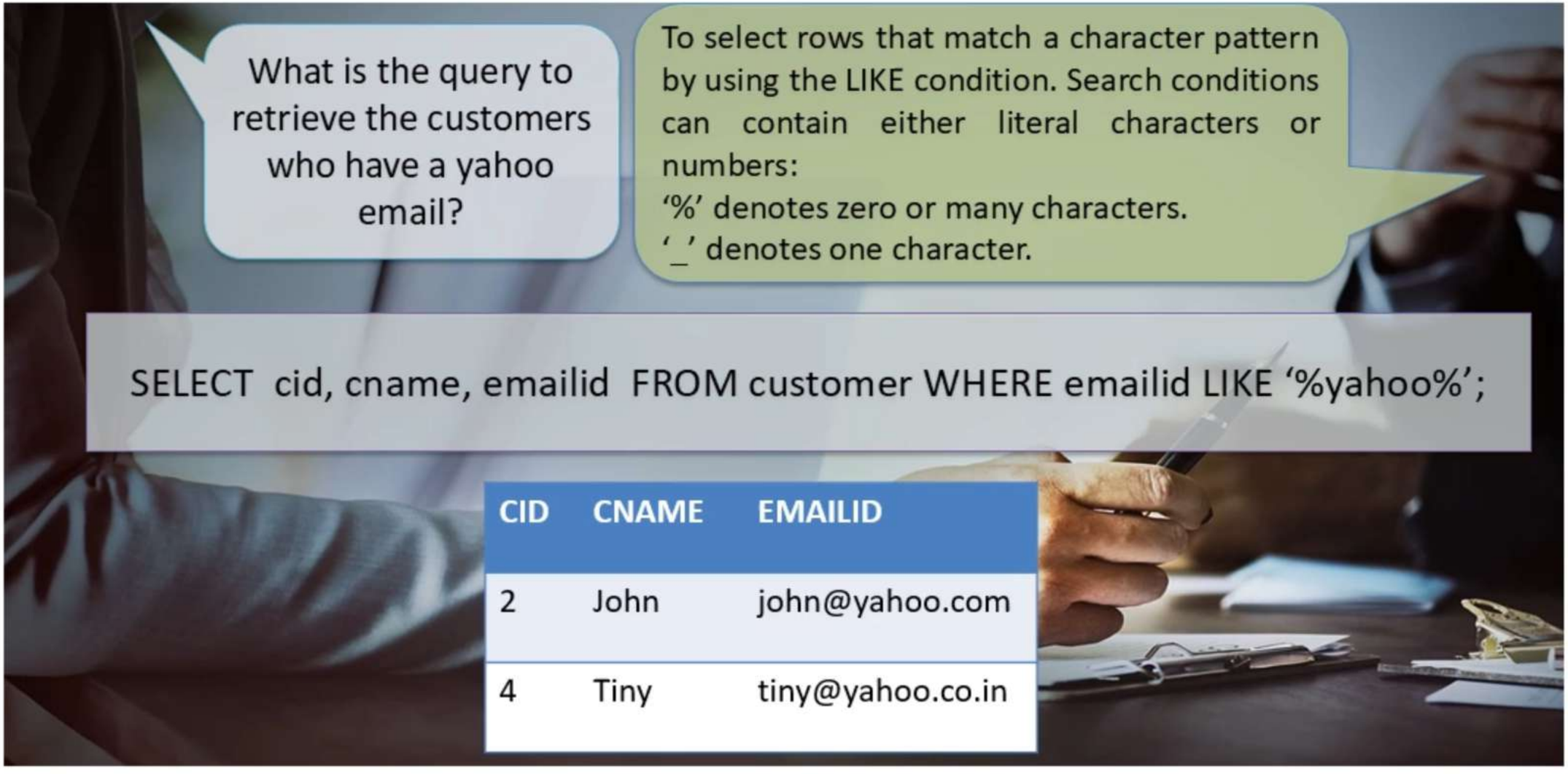
What is the query to retrieve all the policies whose Minimum Amount to be paid is less than 1500?

In the same manner, we can use >=,>,<=

```
SELECT * FROM Policy WHERE MinAmountPerMonth<1500;
```

PID	PNAME	PPERIODINYEARS	MINAMOUNTPERMONTH
MBP	Money Back Plan	20	1000

Example - Like operator

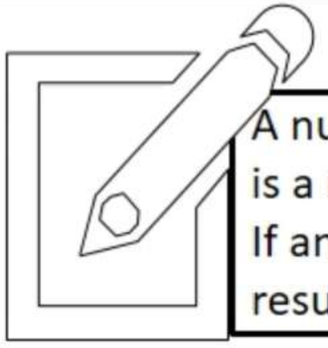


CID	CNAME	EMAILID
2	John	john@yahoo.com
4	Tiny	tiny@yahoo.co.in

WHERE CLAUSE

TEKNOTURF

OPERATORS	MEANING	EXAMPLE	DESCRIPTION
Between X and Y	Range of values between X and Y (X,Y is inclusive)	……WHERE Salary Between 5000 and 8000	List the employees who earn salary between 5000 to 8000 i.e 5000,5001,……7999,8000
IN (set)	Match any of the list values	……WHERE Salary IN (5000,6000,7000)	List the employees who earn salary of 5000 or 6000 or 7000 exactly
IS NULL	Is a null value	……WHERE Salary IS NULL	List the employees whose salary is empty



A null is not the same as zero or a space. Zero is a number, and a space is a character.
If any operation has NULL value, then the result is null.

Example – Between, In, Is Null

To retrieve policy details where minimum amount to be paid is between 2500 and 5000.

```
SELECT * FROM policy WHERE MinAmountPerMonth BETWEEN 2500 AND 5000;
```

To retrieve customer id and policy id of customers whose customer id is 101,105

```
SELECT cid, pid FROM policyenrollment WHERE cid IN (101,105);
```

To retrieve policy enrollment details of those who have not paid penalty for late payment

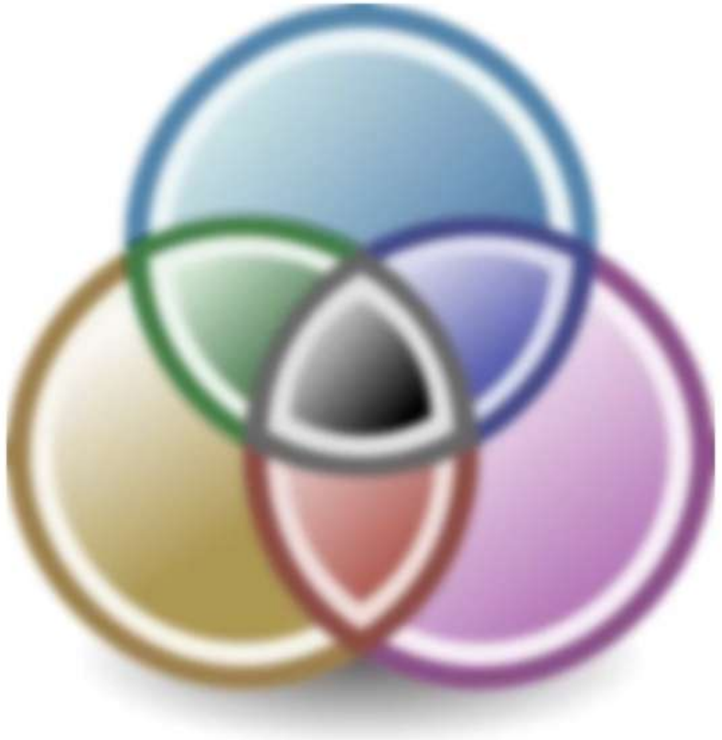
```
SELECT  * FROM policyenrollment WHERE penalty IS NULL;
```

Logical operators

OPERATOR	MEANING
AND	Returns true if both the conditions are true.
OR	Returns true if either of the condition is true.
NOT	Returns true if the condition is false.

```
SELECT  cid, cname FROM customer WHERE cid=1 AND
address='chennai';
```

'NOT' can be used with BETWEEN, IN, LIKE and IS NULL operators.



Order by clause

TEKNOTURF

To retrieve names of the customers in ascending order.

```
SELECT  Cname  FROM customer ORDER BY cname;
```

CNAME
John
Mini
Ram
Tiny
Tom

Default - ascending order.

For descending order:
Select cname from customer **order by cname desc**;

Summary

- Select Statement
- Where Clause
- Types of Operators
- Order by Clause
- Distinct

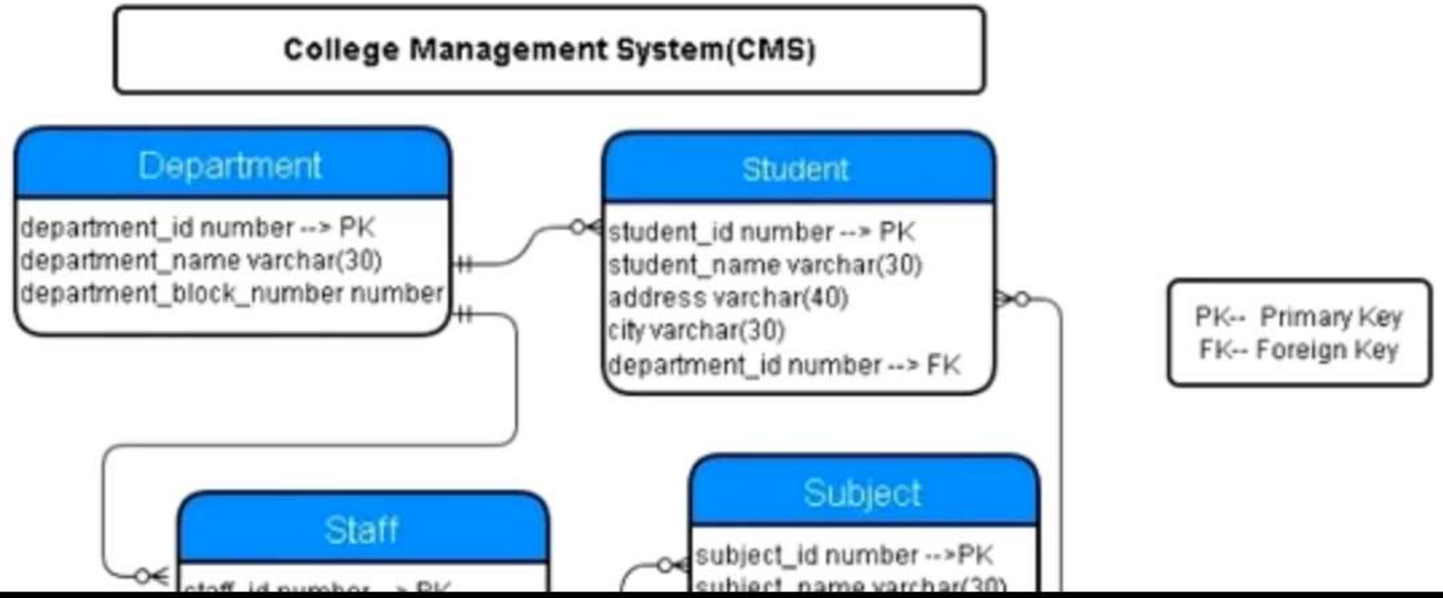
Description Edit Grading view

List Staff Details

Requested files: bselect1.sql
(Download)

Type of work: Individual work
Write a query to display all staff details and sort the result by ID in ascending order.

Refer the below schema diagram:



Description Edit Grading view

File List Save Compile & Run Evaluate Full screen

Save (Ctrl-S)

bselect1.sql

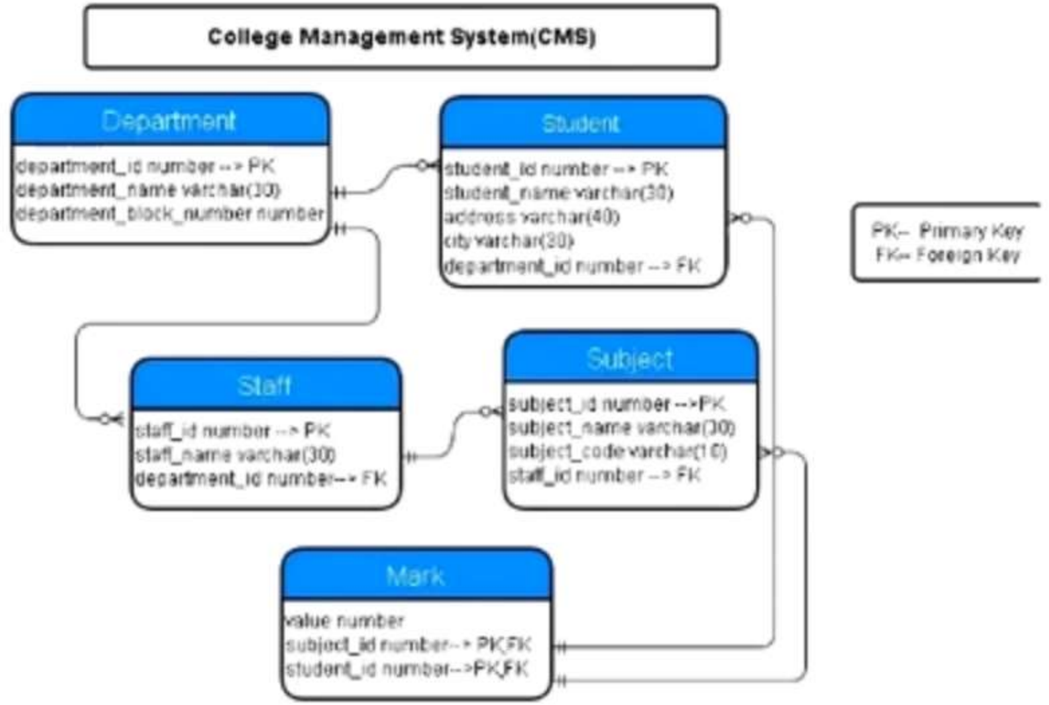
```
1 select * from staff order by staff_id;
```

forms/edit.php?id=13231&userid=3303#

Question Description

Write a query to display all staff details and sort the result by ID in ascending order.

Refer the below schema diagram



Description Edit Grading view

File List Save

bselect1.sql

```
1 select * from s
```

>_ Console: connection closed (Running)

```
 STAFF_ID STAFF_NAME                    DEPARTMENT_ID
---------- ------------------------------ -------------
         1 James                                  1
         2 David                                  1
         3 Christopher                            2
         4 George                                 2
         5 Ronald                                 3
         6 John                                   3
         7 Richard                                3
         8 Daniel                                 4
         9 Kenneth                                4
        10 Anthony                                5
        11 Robert                                 5
```

Description Edit Grading view

File List | Save | Compile & Run | Evaluate | Full screen | Description

bselect1.sql

```
1 select * from staff order by staff_id;
```

Proposed grade: 100 / 100

Result Description

Summary of tests

```
+-----------------------------+
| 2 tests run / 2 test passed |
+-----------------------------+
```

TEKNOTURF

FUNCTIONS IN SQL

IN THIS MODULE YOU WILL LEARN

TEKNOTURF

- Character Functions
- Number Functions
- Date Functions
- Conversion Functions
- Nesting Functions
- General Functions
- Group Functions
- Group by clause
- Having clause

INSURANCE MANAGEMENT SYSTEM(IMS)

TEKNOTURF

We will be using IMS through out this module

Customer	Policy	PolicyEnrollment
• CId • CName • Phoneno • DOB • Email • Address	• PId • PName • PPeriodInYears • MinAmountPerMonth	• EnrollmentId • CId • PId • DueDate • PaidDate • Amount • Penalty

SAMPLE RECORDS

CID	CNAME	Phoneno	DOB	Emailid	Address
1	Tom	9876523190	17-MAR-87	NULL	chennai
2	John	NULL	26-JAN-86	john@yahoo.com	delhi
3	Ram	7654321890	14-DEC-85	ram@gmail.com	pune
4	Tiny	NULL	28-MAY-86	NULL	chennai

Customer

PId	PName	PPeriodInYears	MinAmountPerMonth
MBP	Money Back Plan	20	1000
PP	Personal Protect	15	1500

Policy

EnrollmentID	CId	PId	DueDate	PaidDate	Amount	Penalty
101	3	MBP	12-Dec-2017	11-Dec-2017	2000	NULL
102	1	PP	15-Mar-2018	13-Mar-2018	3000	NULL
103	2	PP	15-Feb-2018	NULL	4000	200

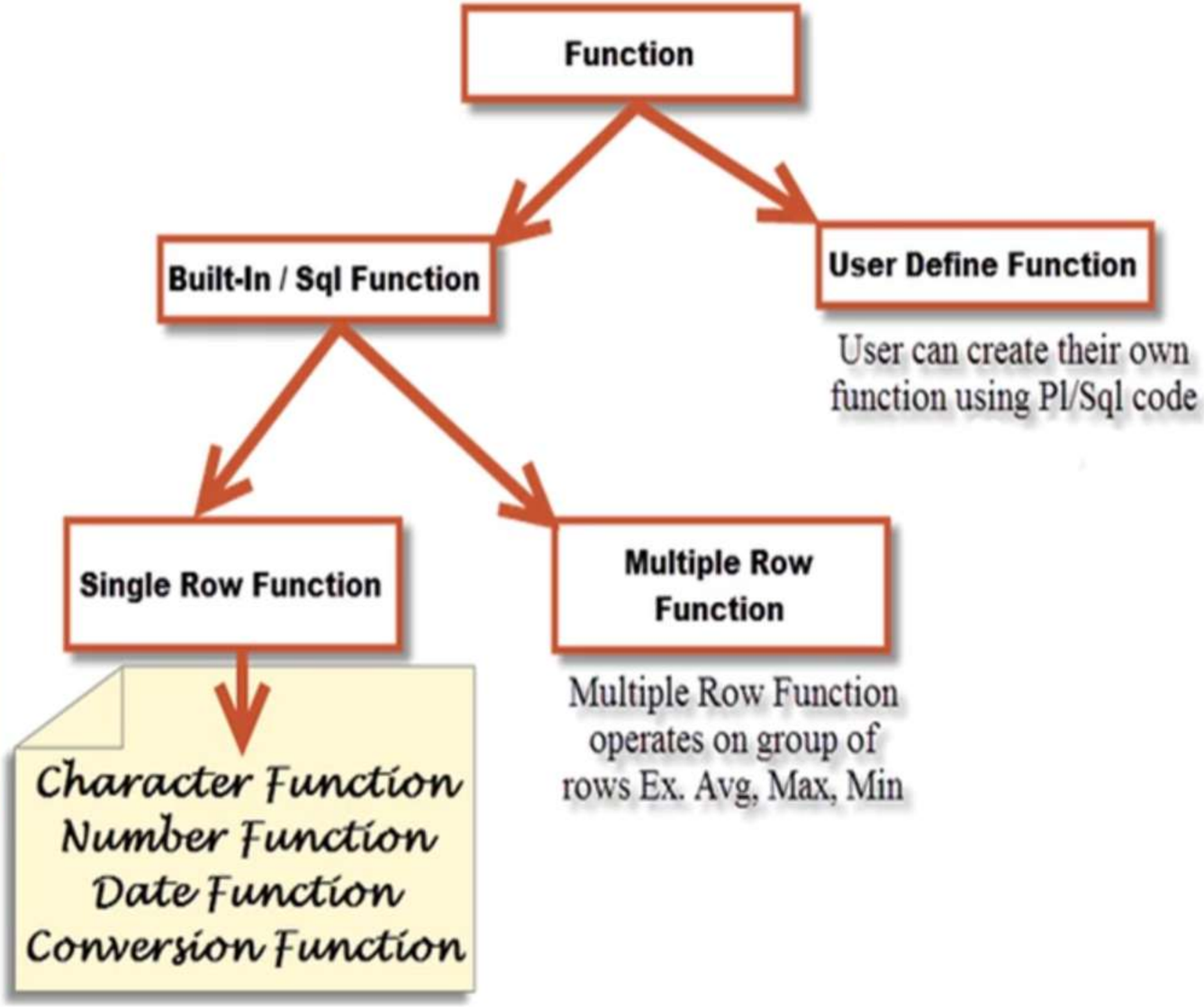
PolicyEnrollment

FUNCTION

A function is a block of code that sometimes take arguments and always returns a value.

SQL functions can be used to do the following:

1. Perform calculations on data.
2. Modify individual data items.
3. Manipulate output for groups of rows.
4. Format dates and numbers for display.
5. Convert column data types.



SINGLE ROW FUNCTION

TEKNOTURF

- Manipulates data items.
- Accepts arguments and returns one value.
- Acts on each row returned.
- Returns one result per row.
- May modify the data type.
- Can be nested.
- Accepts arguments which can be a column or an expression.
- function_name [(arg1, arg2,...)]

CHARACTER FUNCTION

TEKNOTURF

Single-row character functions accept character data as input and

can return both character and numeric values.

Function	Purpose
Lower(column|exp)	Converts alpha character values to lower case.
Upper(column|exp)	Converts alpha character values to upper case.
Initcap(column|exp)	Converts alpha character values to upper case for the first letter of each word.
Concat(col1|exp1,col2|exp2)	Concatenates two columns or expression values.
Substr(col|exp,m,[n])	Returns specified characters from the character value starting at character position m, n characters long
Length(col|exp)	Returns the number of characters in the expression.

CHARACTER FUNCTION - EXAMPLE

TEKNOTURF

Function	Example	Output
Lower(column|exp)	SELECT LOWER(Cname) FROM Customer WHERE cid=1;	Tom
Upper(column|exp)	SELECT UPPER(Cname) FROM Customer WHERE cid=1;	TOM
Initcap(column|exp)	SELECT INITCAP(Cname) FROM Customer WHERE cid=1;	Tom
Concat(col1|exp1,col2|exp2)	SELECT CONCAT (Cname, Address) FROM Customer WHERE cid=1;	Tomchennai
Substr(col|exp,m,[n])	SELECT SUBSTR(Cname,1,2) FROM Customer WHERE cid=1; SELECT SUBSTR(Cname,1) FROM Customer WHERE cid=1; SELECT SUBSTR(Cname,-2,2) FROM Customer WHERE cid=1;	To Tom om
Length(col|exp)	SELECT LENGTH(Cname) FROM Customer WHERE cid=1;	3

NUMBER FUNCTION

TEKNOTURF

Number functions accept numeric input and return numeric values.

Function	Purpose
Abs(col|exp)	returns the absolute value of n.
Ceil(col|exp)	returns the smallest integer value that is greater than or equal to a *number*.
Floor(col|exp)	returns the largest integer value that is equal to or less than a *number*.
Mod(m,n)	returns the remainder of *m* divided by *n*.
Round(n[,m])	returns a number rounded to a certain number of decimal places. m refers to decimal places
Trunc(n[,m]	returns a number truncated to a certain number of decimal places.

NUMBER FUNCTION – EXAMPLE

Function	Example	Output
Abs(col|exp)	SELECT ABS(-5) FROM DUAL;	5
Ceil(col|exp)	SELECT CEIL(5.3) FROM DUAL;	6
Floor(col|exp)	SELECT FLOOR(5.8) FROM DUAL;	5
Mod(m,n)	SELECT MOD(10,2) FROM DUAL;	0
Round(n[,m])	SELECT ROUND(10.678,1) FROM DUAL; SELECT ROUND(10.678) FROM DUAL;	10.7 11
Trunc(n[,m]	SELECT TRUNC(10.678,1) FROM DUAL; SELECT TRUNC(10.67) FROM DUAL;	10.6 10

Dual – dummy table with a varchar data type column

DATE FUNCTION

TEKNOTURF

To get current system date

SELECT SYSDATE from Dual;

Arithmetic operation

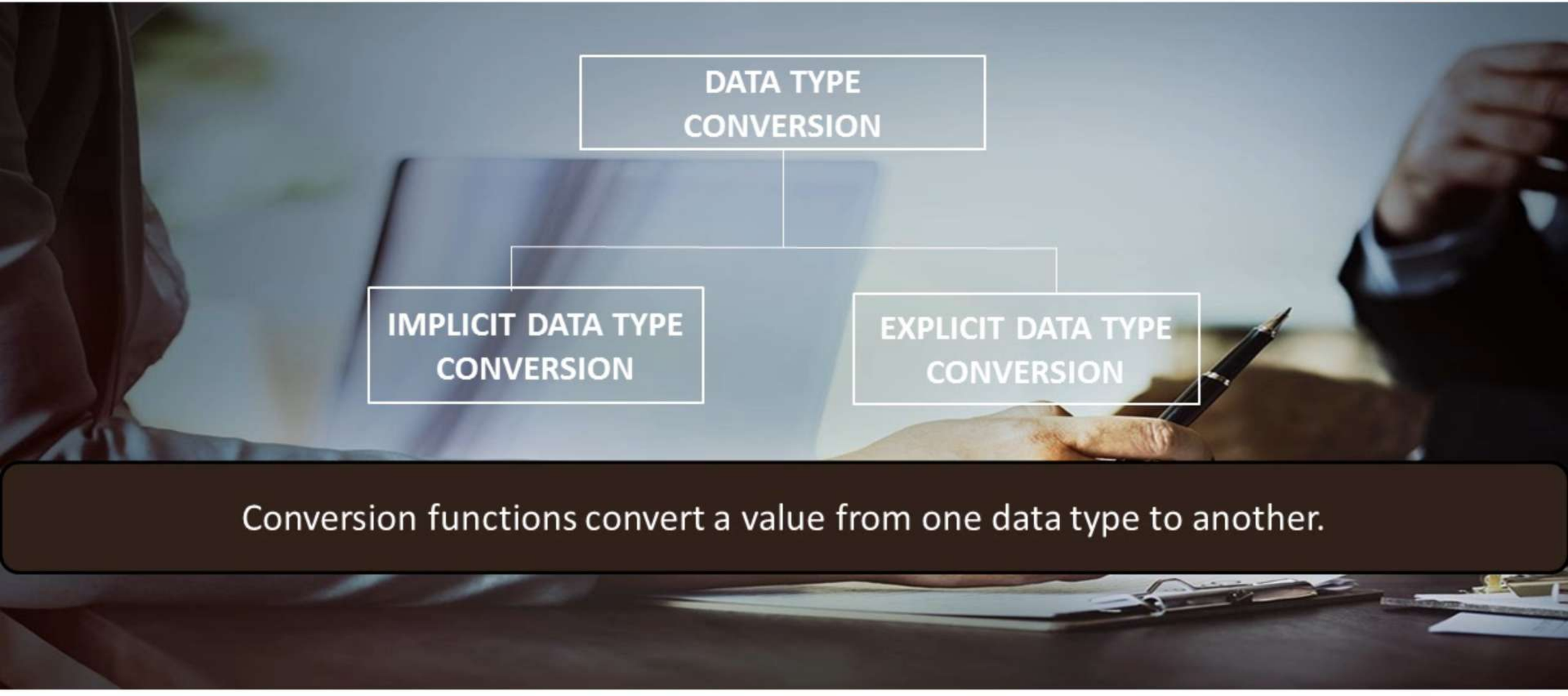
- Add days to date.
- Subtract days from date.
- Subtract two dates.

DATE FUNCTION

TEKNOTURF

Function	Purpose	Example	Output
Months_between	Number of months between two dates	SELECT cid, cname, ROUND(months_between (sysdate, dob)/12) age FROM customer WHERE cid=1;	25
Add_months	Add calendar months to date	SELECT ADD_MONTHS(SYSDATE,4) FROM DUAL; -- current date : 18-APR-19	18-AUG-19
Next_date	Next date of the day specified	SELECT NEXT_DAY(SYSDATE,'monday') FROM DUAL;	23-APR-19
Last_day	Last day of the month	SELECT LAST_DAY(SYSDATE) FROM DUAL;	30-APR-19
Round	Round date	SELECT ROUND(SYSDATE,'year') FROM DUAL; SELECT ROUND(SYSDATE,'month') FROM DUAL;	01-JAN-19 01-MAY-19
Trunc	Truncate date	SELECT TRUNC(SYSDATE,'year') FROM DUAL; SELECT TRUNC(SYSDATE,'month') FROM DUAL;	01-JAN-19 01-APR-19

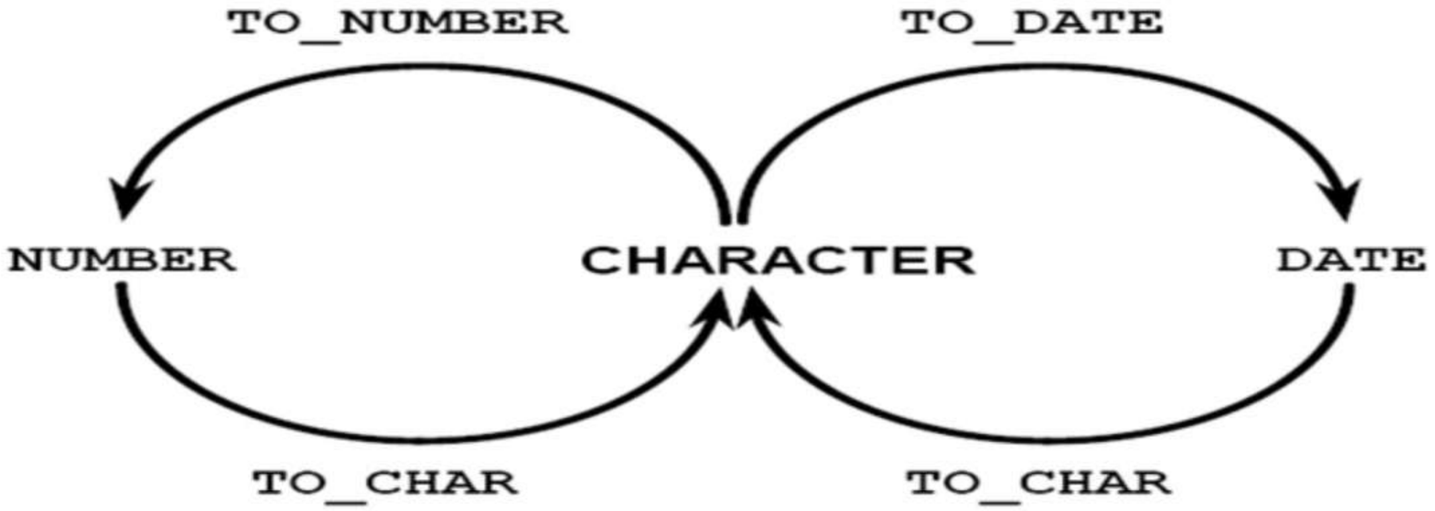
CONVERSION FUNCTION



Conversion functions convert a value from one data type to another.

CONVERSION FUNCTION

Function	Purpose
To_char(number|date,[fmt])	Converts a number or date to string
To_date(char,[fmt])	Converts a string to a date
To_number(char,[fmt])	Converts a string to a number



ELEMENTS OF THE DATE FORMAT MODEL

Parameter	Explanation
Year	Year, spelled out
YYYY	4-digit year
YY	2-digit year
MM	Month number(1-12)
MONTH	Abbreviated name of month
MON	Full name of month
WW	Week of year
W	Week of month
D	Day of week
DAY	Name of day
DD	Day of month
DDD	Day of year
HH	Hour of day(0-12)
HH24	Hour of day(0-23)
MI	Minute
SS	Second

CONVERSION FUNCTION - EXAMPLE

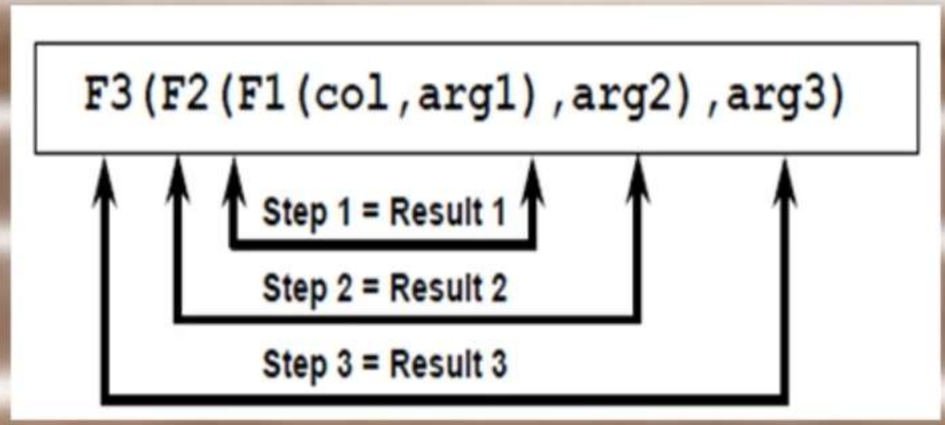
TEKNOTURF

FUNCTION	EXAMPLE	OUTPUT
TO_CHAR(number|date,[fmt])	SELECT TO_CHAR(SYSDATE,'dd month yyyy') FROM DUAL;	18 APRIL 2019
TO_DATE(char,[fmt])	SELECT TO_DATE('may-12-2019','mon-dd-yyyy') FROM DUAL;	12-MAY-19
TO_NUMBER(char,[fmt])	SELECT TO_NUMBER('234.67',999.99) FROM DUAL;	234.67

NESTING FUNCTIONS

TEKNOTURF

- Single-row functions can be nested to any level.
- Nested functions are evaluated from deepest level to the least deep level.




```
select cid, cname, round(months_between (sysdate, dob)/12) age from customer where cid=1;
```

CONDITIONAL EXPRESSION

TEKNOTURF

Gives the use of IF-THEN-ELSE logic within a SQL statement.

Two methods

- CASE expression.
- DECODE function.

CASE EXPRESSION

TEKNOTURF

Facilitates conditional inquiries by doing the work of an IF-THEN-ELSE statement.

```
SELECT pid,pname,
CASE pname
WHEN 'Money Back Plan' THEN 'Money Savings'
WHEN 'Personal Protect' THEN ‘Life Insurance'
ELSE ‘Normal Policy'
END “Category of Policies" FROM policy;
```

PID	PNAME	CATEGORY OF POLICIES
MBP	Money Back Plan	Money Savings
PP	Personal Protect	Life Insurance

DECODE FUNCTION

TEKNOTURF

DECODE is a function in Oracle and is used to provide if-then-else type of logic to SQL. It is not available in MySQL or SQL Server.

```
SELECT pid,pname,DECODE(pname, 'Money Back Plan', 'Money Saving', 'Personal Protect', 'Life Insurance ','Normal Policy') "Category of Policies" FROM Policy;
```

PID	PNAME	CATEGORY OF POLICIES
MBP	Money Back Plan	Money Savings
PP	Personal Protect	Life Insurance

GENERAL FUNCTION

TEKNOTURF

These functions work with any data type and pertain to using null value.

FUNCTION	PURPOSE
NVL(exp1,exp2)	Converts a null value to an actual value
NVL2(exp1,exp2,exp3)	If exp1 is not null then return exp2. If exp2 is null then return exp3
NULLIF	Compares two expressions and returns null if they are equal or the first expression if they are not equal
COALESCE	Returns the first not null expression in the expression list

NVL FUNCTION

- Converts a null to an actual value.
- Data types that can be used are date, character, and number.
- Data types must match.

```
SELECT cid, pid, duedate,paiddate,amount,NVL(penalty,0) penalty FROM
policyenrollment;
```

CID	PID	DUEDATE	PAIDDATE	AMOUNT	PENALTY
3	MBP	12-Dec-2017	11-Dec-2017	2000	0
1	PP	15-Mar-2018	13-Mar-2018	3000	0
2	PP	15-Feb-2018	22-Feb-2018	4000	200

NVL2 FUNCTION

```
SELECT cid, pid, duedate,NVL2(Paiddate,'Paid','Not Paid') status FROM policyenrollment;
```

CID	PID	DUEDATE	STATUS
3	MBP	12-Dec-2017	Paid
1	PP	15-Mar-2018	Paid
2	PP	15-Feb-2018	Not Paid

COALESCE FUNCTION

```
SELECT cid,cname,COALESCE(emailid,TO_CHAR(phoneno),address) contact FROM customer;
```

CID	CNAME	CONTACT
1	Tom	9876523190
2	John	john@yahoo.com
3	Ram	ram@gmail.com
4	Tiny	chennai

AGGREGATE OR GROUP FUNCTION

Group functions operate on the entire set of rows on the table and give one result for the entire set.

Group functions can be nested to a depth of two.

Various group functions

- MAX
- MIN
- SUM
- AVG
- COUNT

Group Functions ignore the null values.

AGGREGATE FUNCTION - EXAMPLE

TEKNOTURF

I want to know what is the maximum and minimum penalty amount paid for late payment. Is there any way to get this information?

You want to get a result from a group of records, isn't it? You can use aggregate function to get the result as you expected.

```
SELECT MAX(penalty) as maximum, MIN(penalty) as minimum FROM
policyenrollment;
```

MAXIMUM	MINIMUM
200	0

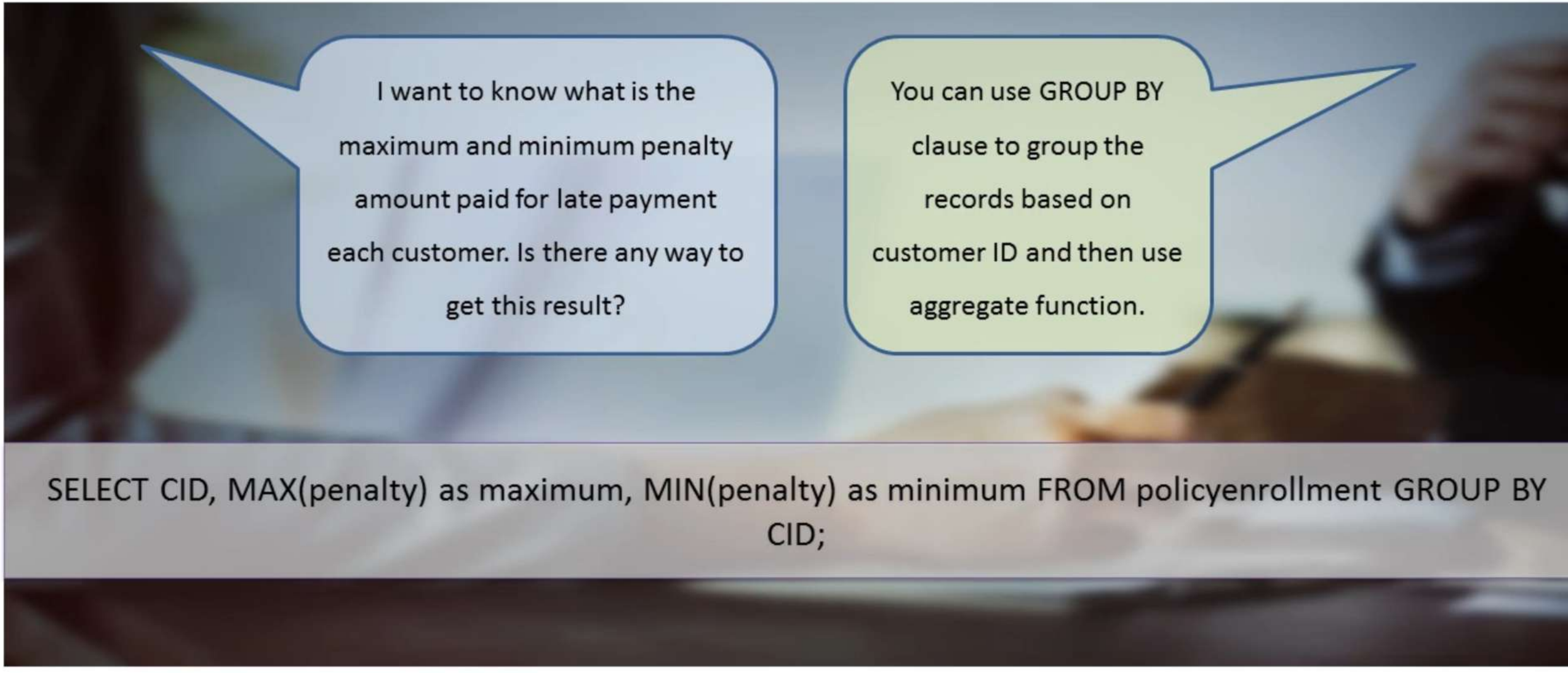
GROUP BY CLAUSE

TEKNOTURF

- The GROUP BY clause divides the rows in a table into groups.
- Divide row of a table into smaller groups by using the GROUP BY clause and with this GROUP BY a column function results in a single value for each group.
- If a group function is included in a SELECT clause along with other non grouped columns, then these non grouped column's should appear in the GROUP BY clause.
- SQL Compiler generates an error if the non grouped columns are not included in the GROUP BY clause.
- The GROUP BY column does not have to be in the SELECT list.
- A column alias cannot be used in the GROUP BY clause.

GROUP BY - EXAMPLE

TEKNOTURF



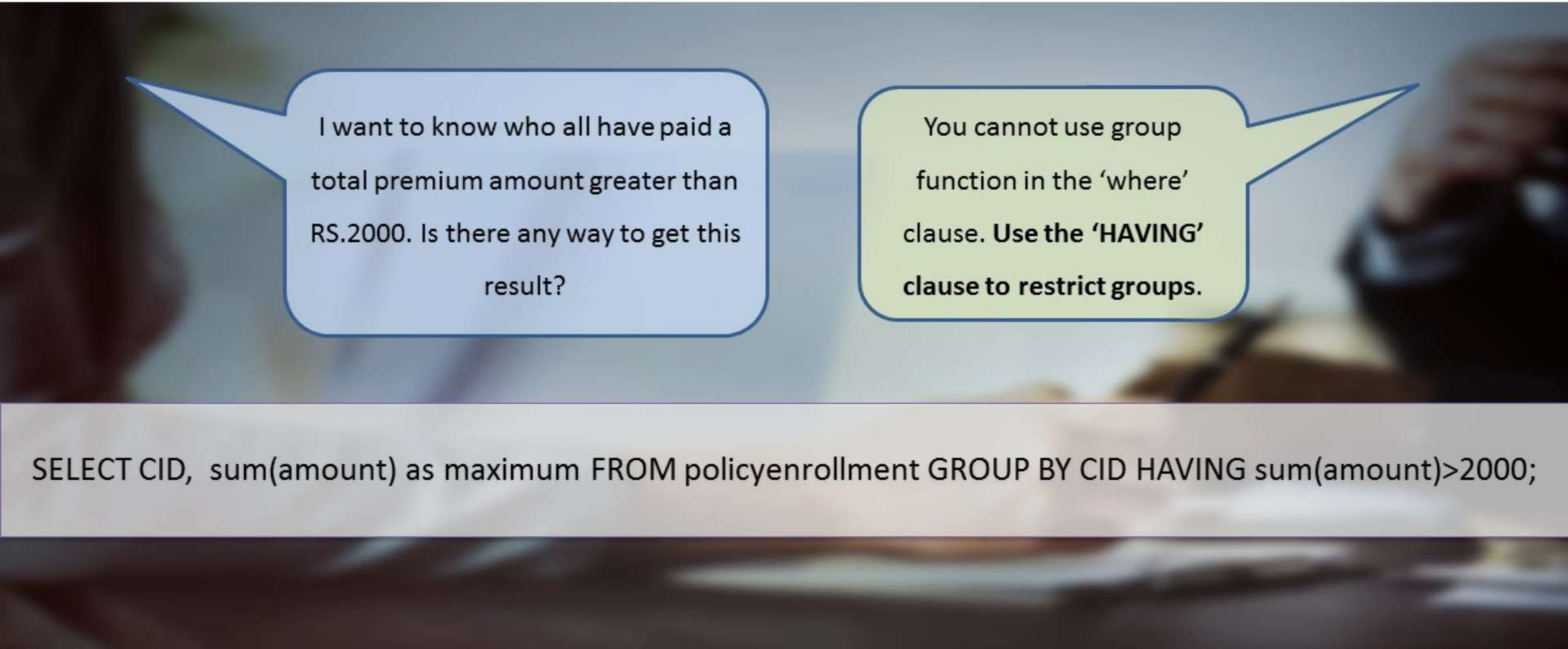

```
SELECT CID, MAX(penalty) as maximum, MIN(penalty) as minimum FROM policyenrollment GROUP BY CID;
```

HAVING CLAUSE

TEKNOTURF

- The WHERE clause cannot be used to restrict groups. The group functions cannot be in the WHERE clause.
- Conditions for groups must be specified in the HAVING clause. HAVING clause of select groups satisfy certain conditions.
- HAVING can specify any column function on any column in a table being queried. This column needs not be in the SELECT list.

Having clause




```
SELECT CID,  sum(amount) as maximum FROM policyenrollment GROUP BY CID HAVING sum(amount)>2000;
```

Summary

- Character Functions
- Number Functions
- Date Functions
- Conversion Functions
- Nesting Functions
- General Functions
- Group Functions
- Group by clause
- Having clause

Description Edit Grading view

Student address

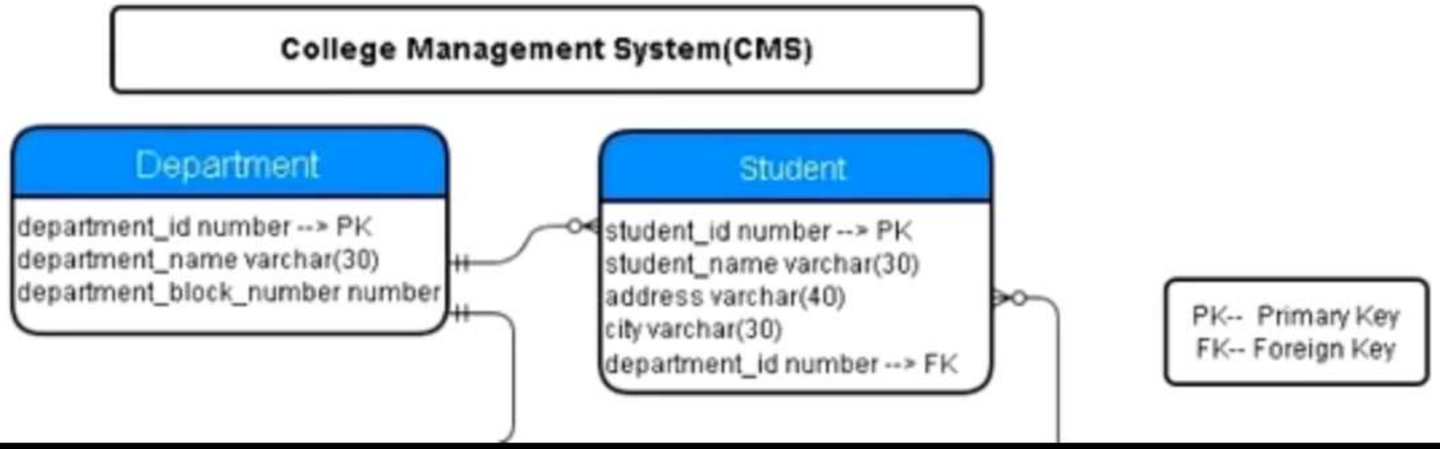
Requested files: SingleRow1.sql
(Download)

Type of work: Individual work

Write a query to display the student id and address for the student "David".

Note : The student name can be in any case.

Refer the below schema diagram:



Description Edit Grading view

File List | Save | Compile & Run | Evaluate | Full screen | Description

SingleRow1.sql

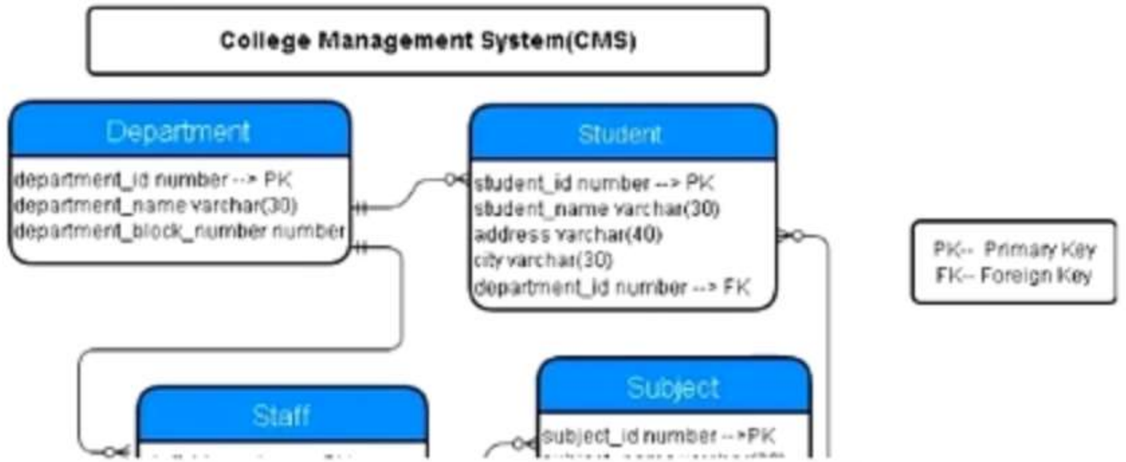
```
1  select student_id,address from
2      student
3      where initcap(student_name)='David';
```

Question Description

Write a query to display the student id and address for the student "David".

Note : The student name can be in any case.

Refer the below schema diagram.



Description Edit Grading view

File List Save

SingleRow1.sql

```
1 select student_
2     student
3     where init
```

>_ Console: connection closed (Running)

```
STUDENT_ID ADDRESS
---------- ----------------------------------------
         3 Downing Street
```

Description Edit Grading view

File List | Save | Compile & Run | Evaluate | Full screen | Description

SingleRow1.sql

```
1  select student_id,address from
2      student
3      where initcap(student_name)='David';
```

Proposed grade: 100 / 100

Result Description

Summary of tests

```
+-----------------------------+
| 2 tests run / 2 test passed |
+-----------------------------+
```

Description Edit Grading view

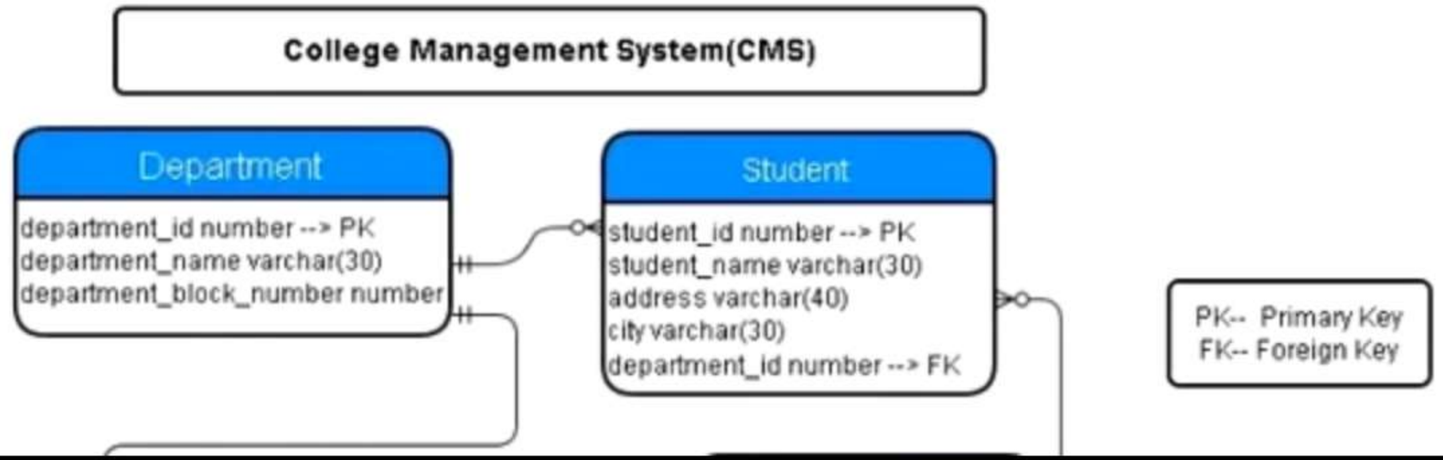
Average mark greater than 80

Requested files: group5.sql
(Download)

Type of work: Individual work

Write a query to display list of student ids and average mark in 2 decimal places if their average mark is greater than 80. Give an alias to average mark as avg_mark. Sort the result based on average mark.

Refer the below schema diagram:



Description Edit Grading view

File List | Save | Compile & Run | Evaluate | Full screen | Description

Run (Ctrl-F11)

group5.sql

```
1  select student_id,round(avg(value),2) as avg_mark
2      from mark
3      group by student_id
4      having (avg(value))> 80;
```

forms/edit.php?id=13443&useri...

Description Edit Grading view

File List Save

group5.sql

```
1 select student_
2     from mark
3     group by st
4     having (avg
```

>_ Console: connection closed (Running: 1 seg)

```
STUDENT_ID   AVG_MARK
---------- ----------
         1         84
         4      81.67
         7      85.67
```

Description Edit Grading view

File List | Save | Compile & Run | Evaluate | Full screen | Description

group5.sql

```
1  select student_id,round(avg(value),2) as avg_mark
2      from mark
3      group by student_id
4      having (avg(value))> 80;
```

Proposed grade: 0 / 100

Result Description

STUDENT_ID	AVG_MARK
1	84
4	81.67
7	85.67

--- Expected output ---

STUDENT_ID	AVG_MARK
4	81.67
1	84
7	85.67

---Difference in Output---

STUDENT_ID	AVG_MARK
1	84
4	81.67
1	84
7	85.67

Description Edit Grading view

File List | Save | Compile & Run | Evaluate | Full screen | Description

Save (Ctrl-S)

group5.sql

```
1  select student_id,round(avg(value),2) as avg_mark
2      from mark
3      group by student_id
4      having (avg(value))> 80
5      order by avg_mark;
```

Proposed grade: 0 / 100

Result Description

```
STUDENT_ID   AVG_MARK
---------- ----------
         1         84
         4      81.67
         7      85.67
```

--- Expected output ---

```
STUDENT_ID   AVG_MARK
---------- ----------
         4      81.67
         1         84
         7      85.67
```

---Difference in Output---

```
STUDENT_ID   AVG_MARK
---------- ----------
         1         84
         4      81.67
         1         84
         7      85.67
```

forms/edit.php?id=13443&useri...

Description Edit Grading view

File List | Save | Compile & Run | Evaluate | Full screen | Description

group5.sql

```
1 select student_id,round(avg(value),2) as avg_mark
2   from mark
3   group by student_id
4   having (avg(value))> 80
5   order by avg_mark;
```

Proposed grade: 100 / 100

Result Description

Summary of tests

```
+-----------------------------+
| 2 tests run / 2 test passed |
+-----------------------------+
```

TEKNOTURF

JOINS & SUBQUERY

IN SQL

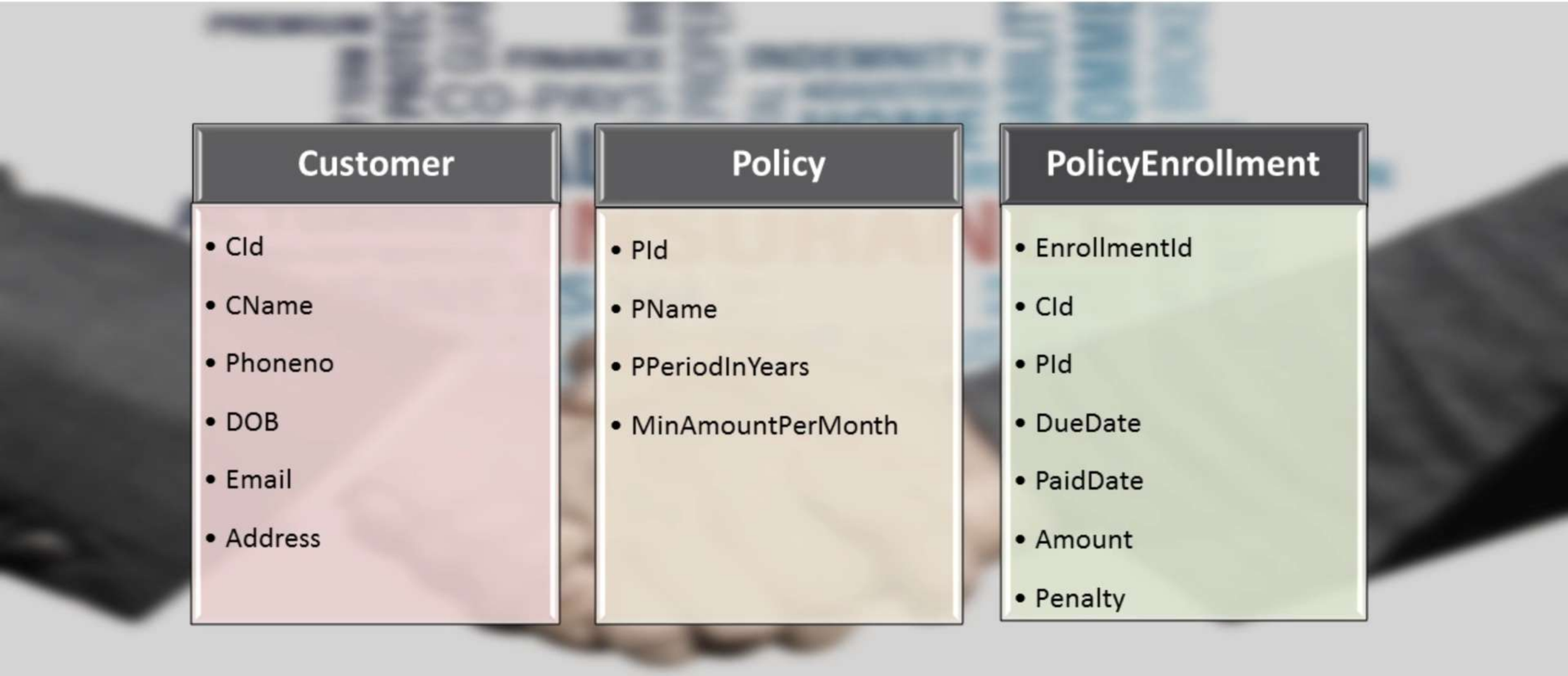
TEKNOTURF

IN THIS MODULE YOU WILL LEARN

- Joins
- Cartesian product
- Types of join
- Natural join
- Subquery
- Single row subquery
- Multirow subquery
- Correlated subquery

INSURANCE MANAGEMENT SYSTEM(IMS)

TEKNOTURF



Customer
• CId
• CName
• Phoneno
• DOB
• Email
• Address

Policy
• PId
• PName
• PPeriodInYears
• MinAmountPerMonth

PolicyEnrollment
• EnrollmentId
• CId
• PId
• DueDate
• PaidDate
• Amount
• Penalty

SAMPLE RECORDS

TEKNOTURF

CID	CNAME	PHONENO	DOB	EMAILID	ADDRESS
1	Tom	9876523190	17-MAR-87	tom@gmail.com	chennai
2	John	8765432190	26-JAN-86	john@yahoo.com	delhi
3	Ram	7654321890	14-DEC-85	ram@gmail.com	pune
4	Tiny	9012365478	28-MAY-86	NULL	chennai

CUSTOMER

PID	PNAME	PPERIODINYEARS	MINAMOUNTPERMONTH
MBP	Money Back Plan	20	1000
PP	Personal Protect	15	1500

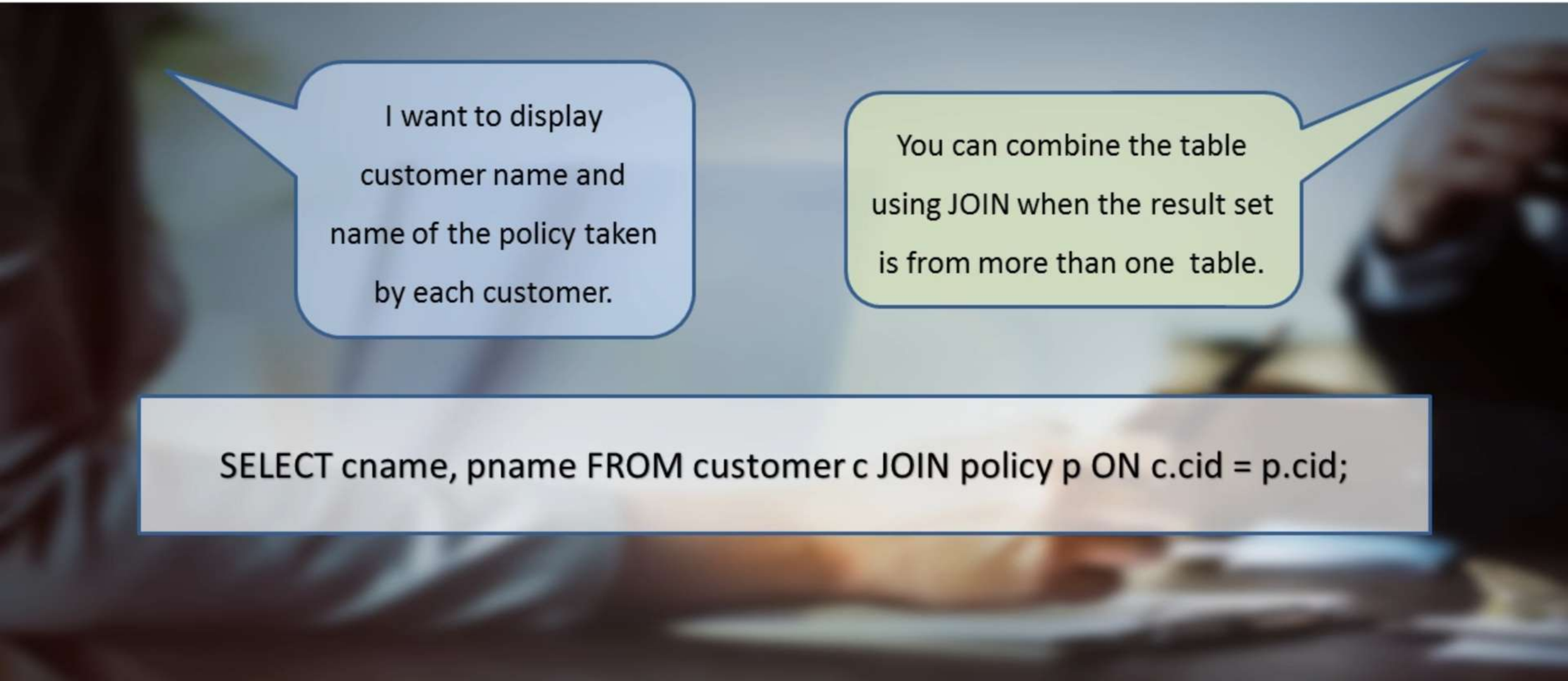
POLICY

ENROLLMENTID	CID	PID	DUEDATE	PAIDDATE	AMOUNT	PENALTY
101	3	MBP	12-Dec-2017	11-Dec-2017	2000	0
102	1	PP	15-Mar-2018	13-Mar-2018	3000	0
103	2	PP	15-Feb-2018	22-Feb-2018	4000	200

POLICY ENROLLMENT

JOINS - INTRODUCTION

TEKNOTURF



JOINS

TEKNOTURF

- A join is a query that combines records from two or more tables.
- A join will be performed whenever multiple tables appear in the FROM clause of the query.
- The select list of the query can select any columns from any of these tables.
- If join condition is omitted or invalid then a **Cartesian product** is formed.

- If any two of these tables have a column name in common, then must qualify these columns throughout the query with table or table alias names to avoid ambiguity.
- Most join queries contain at least one join condition, either in the FROM clause or in the WHERE clause.

CARTESIAN JOINS

TEKNOTURF

Display the details of the customers who have taken the Policy.

CUSTOMER

CID	CNAME
1	Tom
2	John
3	Ram

POLICYENROLLEMENT

EnrollmentID	CID	PID
101	3	MBP
102	1	PP
103	2	PP

```
SELECT c.cid, cname, enrollmentID, p.cid, pid
FROM  customer c, policyenrollment p
Order by c.cid, enrollmentid;
```

OUTPUT

CID	CNAME	ENROLLMENTID	CID	PID
1	Tom	101	3	MBP
1	Tom	102	1	PP
1	Tom	103	2	PP
2	John	101	3	MBP
2	John	102	1	PP
2	John	103	2	PP
3	Ram	101	3	MBP
3	Ram	102	1	PP
3	Ram	103	2	PP

CARTESIAN PRODUCT

TEKNOTURF

- To avoid a Cartesian product, always include a valid join condition.

CUSTOMER

CID	CNAME
1	Tom
2	John
3	Ram

POLICYENROLLEMENT

EnrollmentID	CId	PId
101	3	MBP
102	1	PP
103	2	PP

OUTPUT

CID	CNAME	EnrollmentID	CID	PID
1	Tom	102	1	PP
2	John	103	2	PP
3	Ram	102	3	MBP

```
SELECT c.cid, cname, enrollmentID, p.cid,
pid FROM  customer c, policyenrollment p
WHERE c.cid= p.cid
Order by c.cid;
```

JOIN SYNTAX

TEKNOTURF

- Syntax

```
SELECT table1.column, table2.column
FROM table1, table2
WHERE table1.column1 = table2.column2;
```

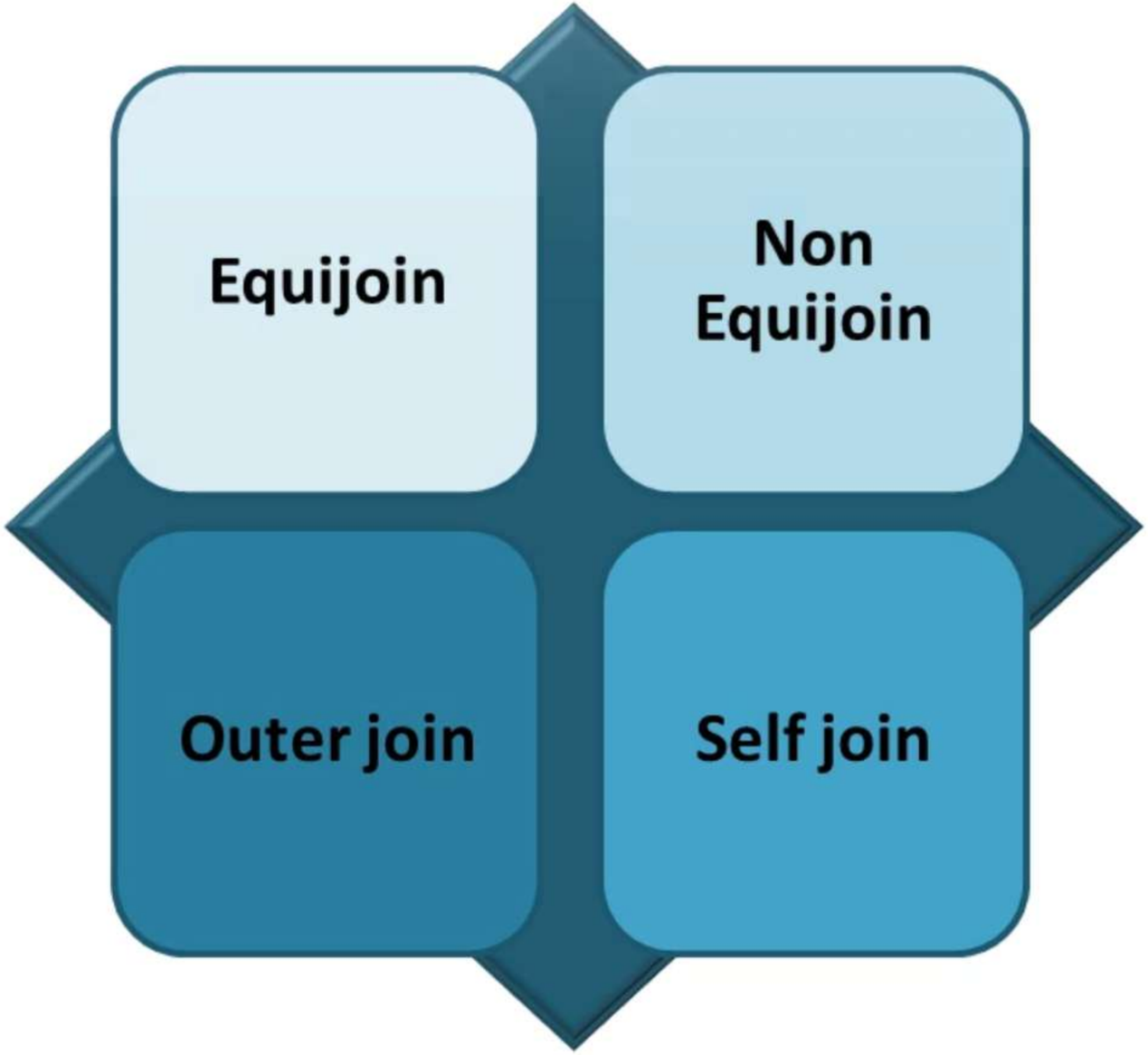
```
SELECT table1.column, table2.column
FROM table1 JOIN table2 ON table1.column1 = table2.column2;
```

ANSI Style join

JOIN and **ON** keyword's are used in the FROM clause of the statement as per **ANSI** standard.

TYPE OF JOINS

TEKNOTURF



EQUIJOIN

TEKNOTURF

What is the join query to display the details of the customers who have taken the Policy?

Equi joins are also called as simple joins or inner joins. When two tables are joined using = operator in the join condition it is called as an equi join.

```
SELECT c.cid, cname, pid FROM customer c, policyenrollment p WHERE c.cid=p.cid;
```

```
SELECT c.cid, cname, pid FROM customer c JOIN policyenrollment p ON c.mid=p.mid;
```

c is the alias name for the table customer and p for policyenrollment

Join condition

A non equijoin is a join condition containing something other than an equality operator.

OUTER JOIN

TEKNOTURF

- Equi join or inner join returns rows, only when there is at least one row from both the tables that matches the join condition.
- Outer joins, return all the rows from one of the tables mentioned in the FROM clause even if the condition is not matched.
- Types of outer join
  - Left outer join
  - Right outer join
  - Full outer join

LEFT OUTER JOIN

TEKNOTURF

- The LEFT OUTER JOIN returns all the rows from the left table, with the matching rows of the right table.
- The result is NULL on the right side when there is no match.

Syntax

```
SELECT table1.column, table2.column
FROM table1 LEFT OUTER JOIN table2 ON
table1.column1 = table2.column2;
```

LEFT OUTER JOIN EXAMPLE

TEKNOTURF

- Display the customer details along with the policy they have taken and also the customers who have not taken any policies.

```
SELECT c.cid, cname, pid FROM customer c LEFT OUTER JOIN policyenrollment p ON c.cid=p.cid;
```

CID	CNAME	PID
1	Tom	PP
2	John	PP
3	Ram	MBP
4	Tiny	

RIGHT OUTER JOIN

TEKNOTURF

- The RIGHT OUTER JOIN returns all the rows from the right table, with the matching rows of the left table.
- The result is NULL in the left side when there is no match.

Syntax

```
SELECT table1.column, table2.column
FROM table1 RIGHT OUTER JOIN table2 ON
table1.column1 = table2.column2;
```

RIGHT OUTER JOIN EXAMPLE

TEKNOTURF

- Display the policies that are taken and not taken by customers

```
SELECT p.pid, pname FROM policyenrollment pp RIGHT OUTER JOIN policy p ON pp.pid=p.pid;
```

PID	PNAME
MBP	Money Back Plan
PP	Personal Protect
MBP	Money Back Plan

FULL OUTER JOIN

TEKNOTURF

- The FULL OUTER JOIN returns all rows from both the tables.
- The result is NULL on the left or right side when there is no match.

Syntax

```
SELECT table1.column, table2.column
FROM table1 FULL OUTER JOIN table2 ON
table1.column1 = table2.column2;
```

FULL OUTER JOIN EXAMPLE

TEKNOTURF

- Display the policies that are taken and not taken. Customers who have taken policies and not taken any policies.

```
SELECT c.cid,cname, p.pid, pname FROM policy p FULL OUTER JOIN policyenrollment pp ON p.pid=pp.pid FULL OUTER JOIN customer c ON pp.cid=c.cid;
```

SELF JOIN

TEKNOTURF

- A self join is a join in which a table is joined with itself (which is also called Unary relationships), especially when the table has a FOREIGN KEY which references it's own PRIMARY KEY.
- To join a table itself means that each row of the table is combined with itself and with every other row of the table.
- The self join can be viewed as a join of two copies of the same table.

Query to display the books which are of the same price.

```
SELECT b1.bid,b2.bname,b2.bprice FROM book b1,book b2 WHERE b1.bid!=b2.bid AND
b1.bprice=b2.bprice;
```

NATURAL JOIN

TEKNOTURF

- NATURAL JOIN is a type of EQUI JOIN that compares the common columns of both the tables with each other.
- The associated tables can have one or more pairs of identically named columns.
- The columns must have the same data type.

Check whether common columns exist in both tables before doing a natural join.

```
SELECT cid, cname, pid FROM customer NATURAL JOIN policyenrollment;
```

JOINS WITH THE USING CLAUSE

TEKNOTURF

The USING clause specifies which columns to test for equality when two tables are joined. It can be used instead of an ON or where clause .

```
SELECT cid, cname, pid FROM customer JOIN policyenrollment USING (cid);
```

JOINING MULTIPLE TABLES

TEKNOTURF

- SELECT table1.column, table2.column,table3.column FROM table1, table2, table3 WHERE table1.column1 = table2.column2 and table1.column1=table3.column1;
- SELECT table1.column, table2.column, table3.column FROM table1 JOIN table2 ON table1.column1 = table2.column2 JOIN table3 ON table1.column1=table3.column1;

To join n number of tables, you need a minimum of n-1 join conditions. for example, to join three tables, two join conditions are required.

JOINING MULTIPLE TABLES - EXAMPLE

TEKNOTURF

- Query to display policies taken by each customer. Display the policy name, customer name and due date.
- You need to join policy, customer and policyenrollment table. The syntax to join multiple table is:

```
SELECT cname,pname,duedate FROM customer c JOIN policyenrollment p ON c.cid=p.cid
JOIN policy pp ON pp.pid=p.pid;
```

SAMPLE RECORDS

TEKNOTURF

CID	CNAME	Phoneno	DOB	Emailid	Address
1	Tom	9876523190	17-MAR-87	tom@gmail.com	chennai
2	John	8765432190	26-JAN-86	john@yahoo.com	delhi
3	Ram	7654321890	14-DEC-85	ram@gmail.com	pune
4	Tiny	9012365478	28-MAY-86	NULL	chennai

Customer

PId	PName	PPeriodInYears	MinAmountPerMonth
MBP	Money Back Plan	20	1000
PP	Personal Protect	15	1500

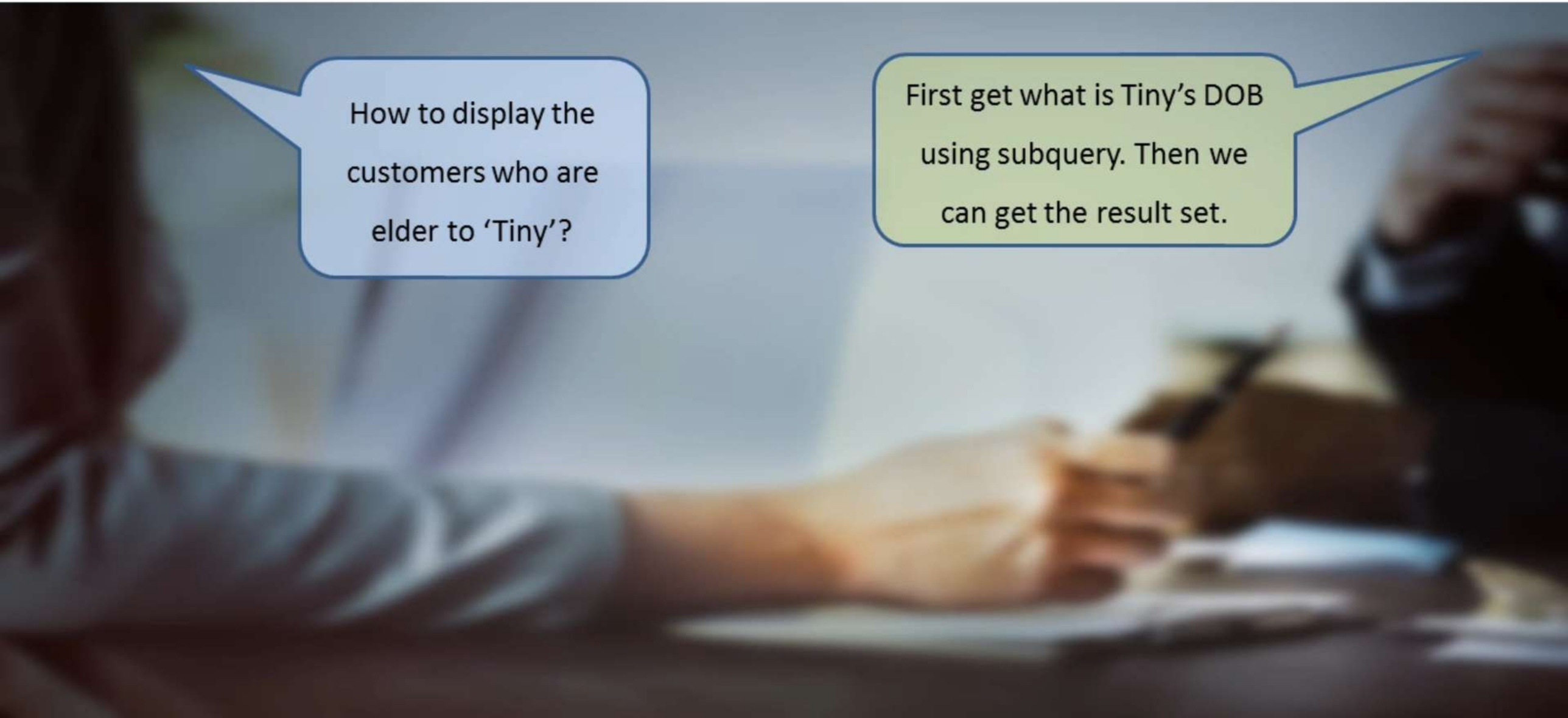
Policy

EnrollmentID	CId	PId	DueDate	PaidDate	Amount	Penalty
101	3	MBP	12-Dec-2017	11-Dec-2017	2000	0
102	1	PP	15-Mar-2018	23-Mar-2018	3000	200
103	2	MBP	22-Apr-2018	27-Apr-2018	4000	120
104	2	PP	15-Feb-2018	22-Feb-2018	4000	200
105	4	MBP	23-Apr-2018	25-Apr-2018	3500	70
106	3	PP	22-Mar-2018	28-Mar-2018	2400	150

PolicyEnrollment

SUBQUERY

TEKNOTURF



SUBQUERY

TEKNOTURF

- A subquery is a SELECT statement that is embedded in a clause of another SQL statement.
- They can be very useful to select rows from a table with a condition that depends on the data in the same or another table.
- The subquery can be placed in the following SQL clauses:

  The WHERE clause.

  The HAVING clause.

  The FROM clause.

SYNTAX for using SUBQUERY in WHERE clause

```
SELECT Column_List
FROM table
WHERE columnname operator
        ( SELECT columnlist
                FROM table ) ;
```

SUBQUERY EXAMPLE

TEKNOTURF

Main query

Query to display the customers who are elder to 'Tiny'

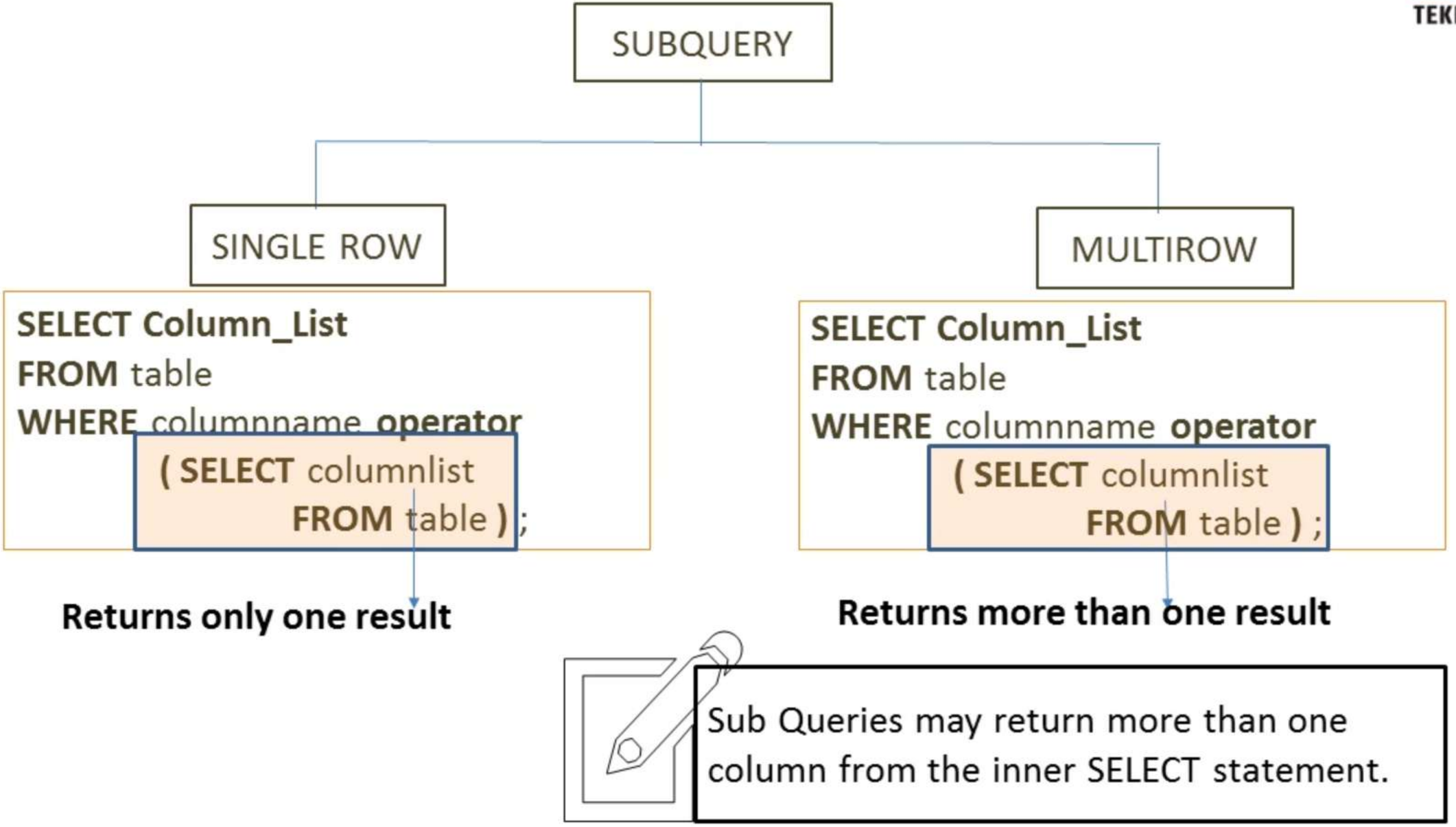
```
SELECT CID,CNAME, DOB from Customer WHERE dob>(SELECT dob FROM
customer  WHERE cname='tiny');
```

Sub query. Enclose sub queries in parentheses.

CID	CNAME	DOB
2	John	27-JUL-85
3	Ram	31-DEC-84

Subquery output is 28-MAY-86. Only output of the main query will be displayed.

SUBQUERIES - TYPES



OPERATORS IN SUBQUERY

TEKNOTURF

Single Row Subquery

- Returns only one row
- Operators used are:

Operator	Meaning
=	Equal to
<	Less than
<=	Less than equal to
>	Greater than
>=	Greater than equal to
<>	Not equal to

Multiple Row Subquery

- Returns more than one row
- Operators used are:

Operator	Meaning
in	Equal to any member in the list
all	Compare to each value returned by the subquery
any	Compare to any value returned by the subquery

MULTIROW SUBQUERY - EXAMPLE

TEKNOTURF

Display the customers who paid the highest penalty

Select cid,cname,address from customer where cid =(select cid from policyenrollment where penalty=(Select max(penalty) from policyenrollment));

1, 2

The sub query returns more than one value, but = operator checks only one value.

200

Use ‘in’ operator, instead of =

Select cid,cname,address from customer where cid IN(select cid from policyenrollment where penalty=(Select max(penalty) from policyenrollment));

MULTI ROW OPERATORS - IN, ANY AND ALL

TEKNOTURF

Combinations of ANY

- <ANY means - less than any one of the available values.
- >ANY means - more than any one of the available values.
- =ANY is equivalent to IN.

Combinations of ALL

- <ALL means less than all available values.
- >ALL means more than all available values.

The NOT operator can be used with IN, ANY, and ALL operators.

ANY OPERATOR - EXAMPLE

TEKNOTURF

- Display the customers who have a greater penalty than the penalty paid by customer 4 or 3(cid).

```
SELECT cid,pid,penalty FROM policyenrollment WHERE penalty >ANY(SELECT penalty from
policyenrollment WHERE mid IN(4,3));
```

CID	PID	PENALTY
1	PP	200
2	MBP	120
2	PP	200

70, 150

ALL OPERATOR – EXAMPLE

TEKNOTURF

- Display the customers who have a greater penalty than the penalty paid by both customers whose ID is 4 and 3.

```
SELECT cid,pid,penalty FROM policyenrollment WHERE penalty >=ALL(SELECT penalty FROM
policyenrollment WHERE mid IN(4,3));
```

CID	PID	PENALTY
1	PP	200
2	PP	200

ALL OPERATOR – EXAMPLE

TEKNOTURF

- Display the customers who have a greater penalty than the penalty paid by both customers whose ID is 4 and 3.

```
SELECT cid,pid,penalty FROM policyenrollment WHERE penalty >=ALL(SELECT penalty FROM
policyenrollment WHERE mid IN(4,3));
```

70, 150

CID	PID	PENALTY
1	PP	200
2	PP	200

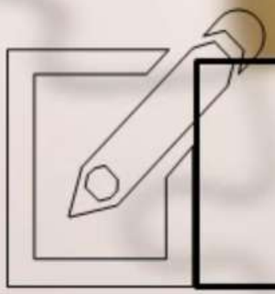
SUBQUERY IN THE SELECT LIST

TEKNOTURF

- Query to display the cid, cname and the number of policies they have taken.

```
SELECT cid,cname,(SELECT count(*) FROM policyenrollment p WHERE p.cid=c.cid) as cnt FROM

customer c;
```



Subquery used SELECT clause must to provided with an alias name.

SUBQUERY AS A TABLENAME

TEKNOTURF

- When you insert a select statement into a FROM clause, it becomes a subquery.
- The subquery returns a temporary table in the database server's memory and then it is used by the outer query for further processing.

- Display the policies that have been taken the maximum number of times.

```
SELECT pid,count(*) maxcount FROM policyenrollment GROUP BY pid
    HAVING COUNT(*)=(SELECT MAX(cnt) maxcnt FROM(SELECT
     pid,count(*) cnt FROM policyenrollment GROUP BY pid));
```

Count(*) will count all the rows

Correlated Subquery

TEKNOTURF

- Correlated subquery is a subquery that uses values from the outer query.
- The subquery is evaluated once for each row processed by the outer query.

- Query to display the policies of the same minimum amount to be paid per month

```
SELECT p.pid,p.pname,p.MinAmount FROM policy p WHERE
p.Minamount=(SELECT p1.minamount FROM policy p1 WHERE
        p1.minamount=p.minamount AND p1.pid!=p.pid);
```

SUBQUERY - INSERT

TEKNOTURF

- All the records in the policyenrollment need to be copied into the policyenrollment_history table.

```
INSERT INTO policyenrollment_history SELECT * FROM policyenrollment;
```

Ensure you have created the policyenrollment_history table before inserting.

SUBQUERY- CREATE

TEKNOTURF

- Is there any way to copy the records without creating the policyenrollment_history table?

CREATE TABLE policyenrollment_history AS SELECT * FROM policyenrollment;

SUBQUERY- DELETE

TEKNOTURF

- What is the query to delete all the policyenrollment made by ‘John’?

```
DELETE FROM policyenrollment WHERE cid=(SELECT cid FROM customer WHERE

cname=‘John’);
```

Summary

- Joins
- Cartesian product
- Types of join
- Natural join
- Subquery
- Single row subquery
- Multirow subquery
- Correlated subquery

TEKNOTURF

Description Edit Grading view

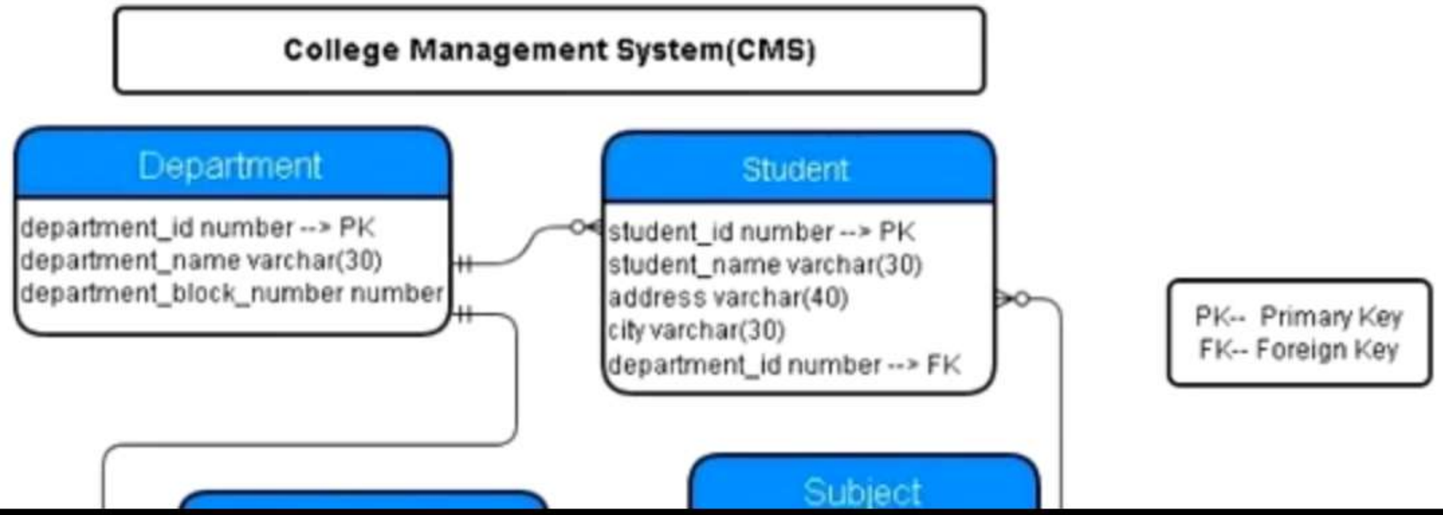
Student and their Department Based on City

Requested files: joins3.sql
(Download)

Type of work: Individual work

Write a query to display list of students name and their department name who all are from Paris. Sort the result based on students name

Refer the below schema diagram:



Description Edit Grading view

File List | Save | Compile & Run | Evaluate | Full screen

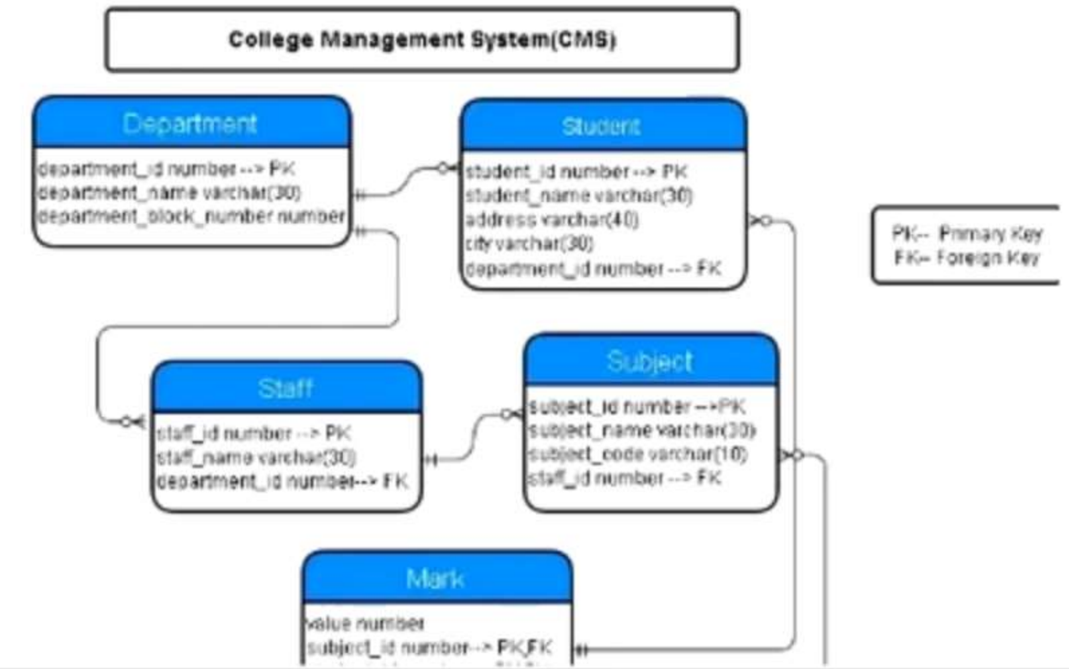
joins3.sql

```
1  select student_name,department_name
2      from department inner join student
3      on student.department_id=department.department_id
4      where lower(city)='paris'
5      order by student_name;
```

Question Description

Write a query to display list of students name and their department name who all are from Paris. Sort the result based on students name

Refer the below schema diagram



Description Edit Grading view

File List Save

joins3.sql

```
1 select student_
2     from depart
3     on student.
4     where lower
5     order by st
```

>_ Console: connection closed (Running: 1 seg)

```
STUDENT_NAME                   DEPARTMENT_NAME
------------------------------ ------------------------------
Austin                         Computer Science
```

Description Edit Grading view

File List | Save | Compile & Run | Evaluate | Full screen | Description

Save (Ctrl-S)

joins3.sql

```
1  select student_name,department_name
2      from department, student
3      where student.department_id=department.department_id and
4      lower(city)='paris'
5      order by student_name;
```

Proposed grade: 100 / 100

Result Description

Summary of tests

```
+-----------------------------+
| 2 tests run / 2 test passed |
+-----------------------------+
```

forms/edit.php?id=13265&useri...

Description Edit Grading view

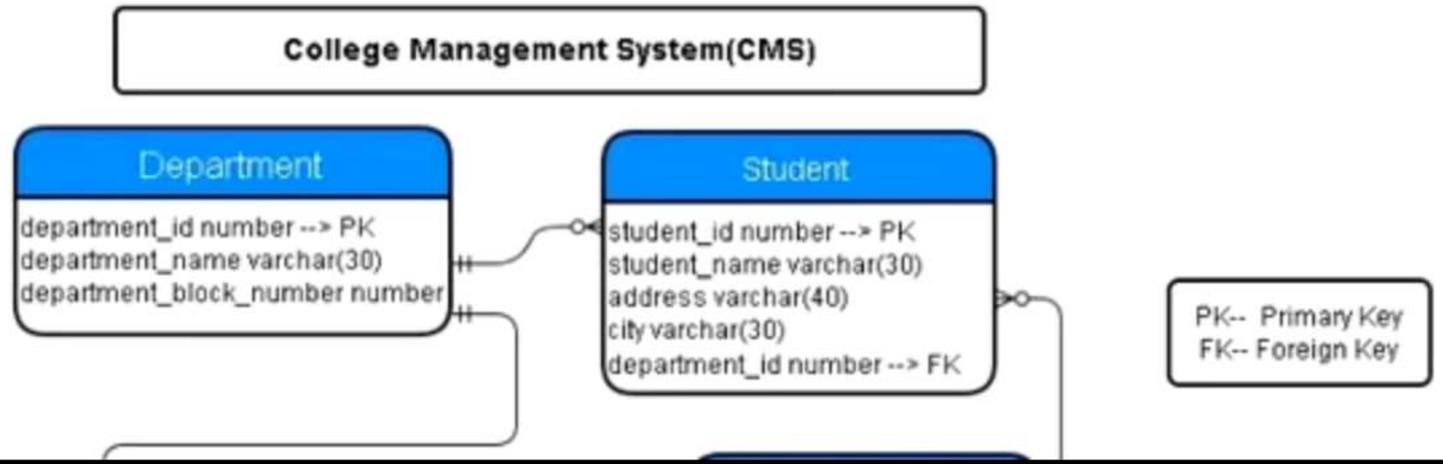
Staff not handling any subject

Requested files: subquery7.sql
(Download)

Type of work: Individual work

Write a query to display the names of the staff who are not handling any subject, ordered by name in ascending order.

Refer the below schema diagram:



Description Edit Grading view

File List Save Compile & Run Evaluate Full screen Description

subquery7.sql

```
1  select staff_name from staff where staff_id not in
2     (select staff_id from subject)
3     order by staff_name;
```

Description Edit Grading view

File List Save

subquery7.sql

```
1 select staff_na
2    (select staf
3    order by sta
```

>_ Console: connection closed (Running: 1 seg)

```
STAFF_NAME
------------------------------
John
Kenneth
Richard
Ronald
```

Description Edit Grading view

File List | Save | Compile & Run | Evaluate | Full screen | Description

subquery7.sql

```
1  select staff_name from staff where staff_id not in
2     (select staff_id from subject)
3     order by staff_name;
```

Proposed grade: 100 / 100

Result Description

Summary of tests

```
+-----------------------------+
| 2 tests run / 2 test passed |
+-----------------------------+
```

TEKNOTURF

DATA CONTROL LANGUAGE
&
DATABASE OBJECTS

In this module you will learn about

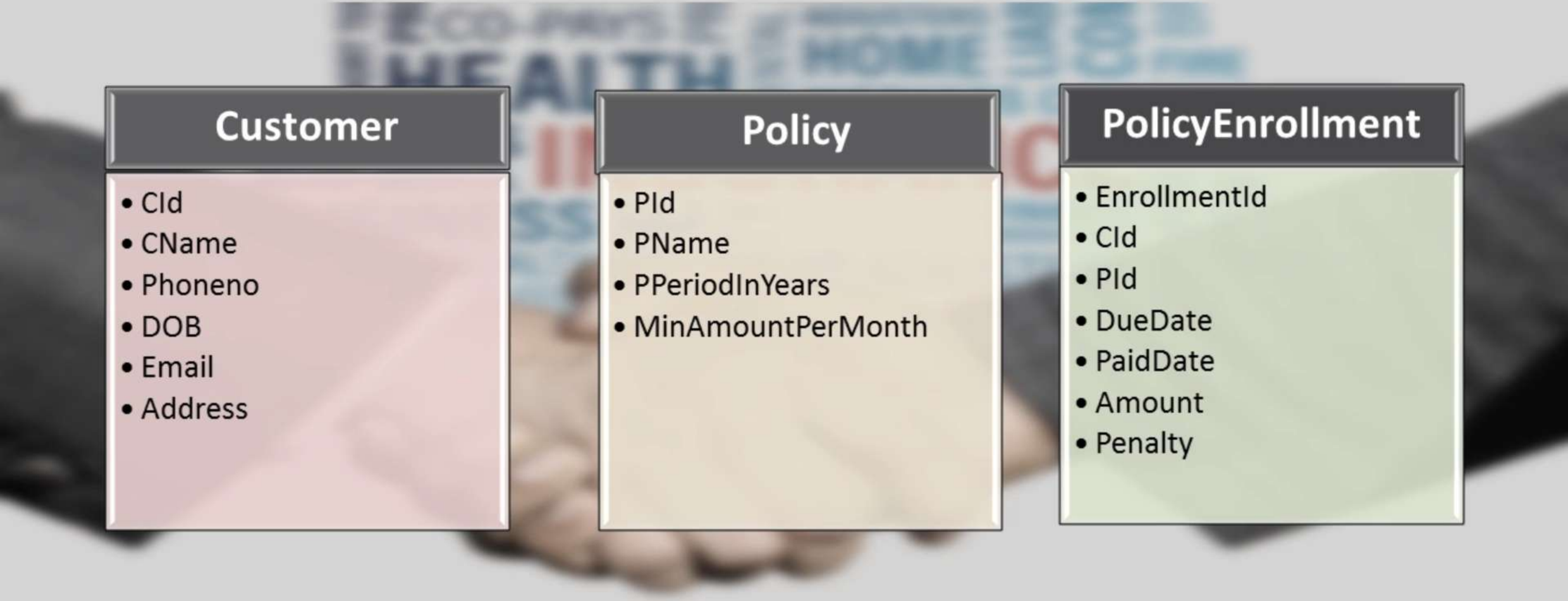
TEKNOTURF

- Grant
- Revoke
- View
- Inline view
- Top N analysis
- Sequence
- Synonym
- Index

Insurance Management System

TEKNOTURF

We will be using IMS through out this module



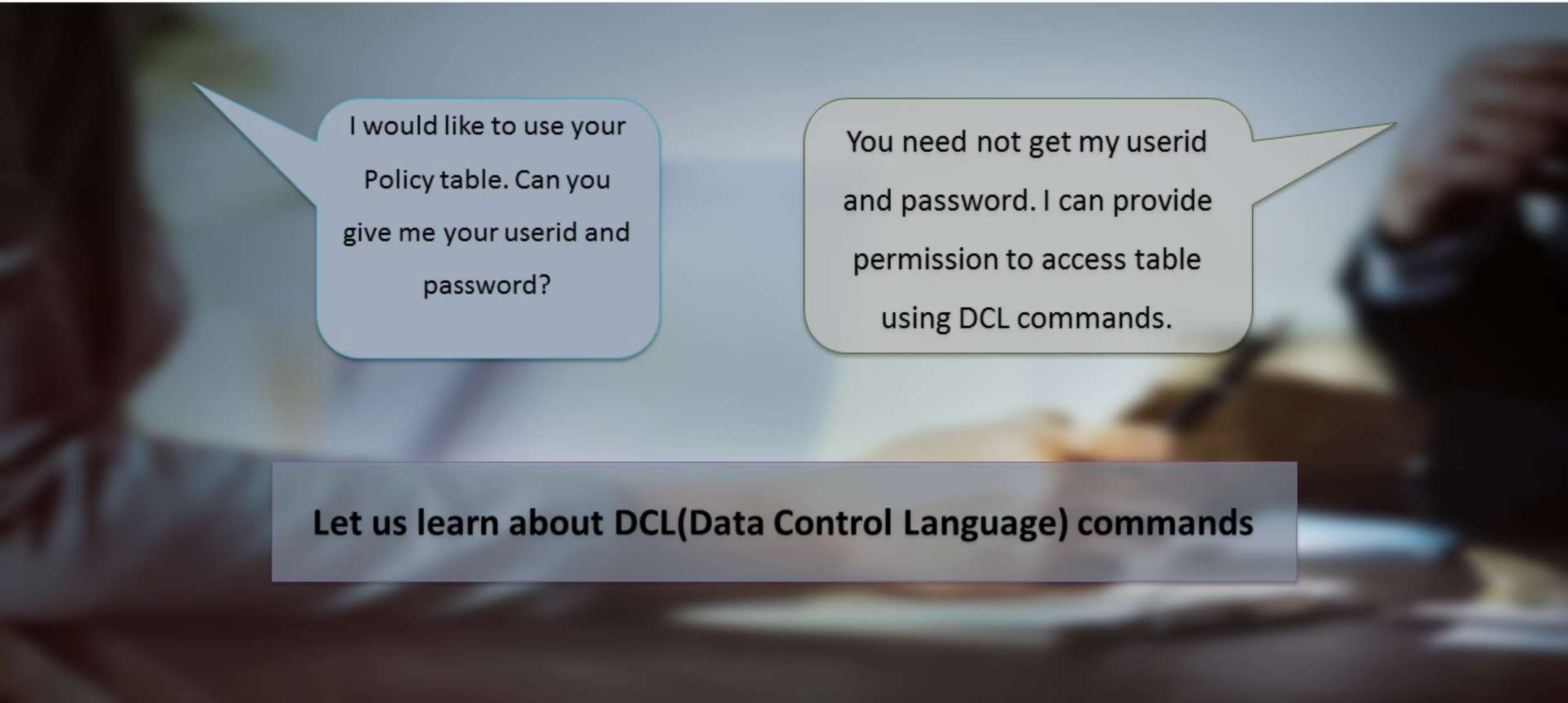
Customer
• CId • CName • Phoneno • DOB • Email • Address

Policy
• PId • PName • PPeriodInYears • MinAmountPerMonth

PolicyEnrollment
• EnrollmentId • CId • PId • DueDate • PaidDate • Amount • Penalty

Overview

TEKNOTURF



Data Control Language

TEKNOTURF

Used to creates privilege to allow users to access and manipulate the database.

Object privileges

- Object privileges vary from object to object
- An owner has all the privileges on the object.
- An owner can give specific privileges on the owner's object

There are two main commands

- GRANT to grant a privilege to the user
- REVOKE to revoke (remove) a privilege from a user

Grant

The GRANT command can be attached to any combination of

SELECT, INSERT, UPDATE, DELETE, ALTER

- GRANT privilege1,privilege2,privilege3 | ALL
  ON TABLE|VIEW
  TO Userid [WITH CHECK OPTION]

Syntax

Example

- GRANT SELECT ON POLICY TO SAM;
- Grant to ALL Users
  - GRANT SELECT ON POLICY TO PUBLIC;

Revoke

TEKNOTURF

Gets back the given permission from the user

Syntax	• REVOKE privilege1,privilege2,privilege3 | ALL ON table|view FROM userId
Example	• REVOKE SELECT ON POLICY FROM sam;

TEKNOTURF


An Oracle database contains multiple Database objects.

OBJECT	DESCRIPTION
Table	Basic unit of storage; composed of rows and columns
View	Logically represents subsets of data from one or more tables
Sequence	Numeric value generator
Index	Improves the performance of some queries
Synonym	Gives alternative names to objects

View

TEKNOTURF

A view is a virtual table that provides a window through which one can see data (stored in a base relation).

Views contain no data of their own but can be operated on as real relations.

Views can help simplify data access by isolating users from querying details.

Syntax

```
CREATE VIEW viewname AS select query [WITH CHECK OPTION [CONSTRAINT
constraint]]  [WITH READ ONLY [CONSTRAINT constraint]];
```

View - Example

TEKNOTURF

Virtual table will point only to the selected records.

PId	PName	PPeriodInYears	MinAmountPerMonth
MBP	Money Back Plan	20	1000
PP	Personal Protect	15	1500

Original table

```
Create view policy_details as select pid,pname
from policy;
```

Query to create virtual table

Querying from view

TEKNOTURF

```
Select * from policy_details;
```

PId	PName
MBP	Money Back Plan
PP	Personal Protect

Deny DML operations

TEKNOTURF

To ensure that no DML operations occur through view, create a view with read only option.

```
Create view policy_details as select pid,pname from policy with read only;
```

We cannot insert, update or delete the records from the original table using view.

View Types

TEKNOTURF

Simple view

- Derives data from only one table
- Contains no functions or groups of data
- Can perform DML operations through the view

Complex view

- Derives data from many tables
- Contains functions or groups of data
- Does not always allow DML operations through the view

Complex view - Example

TEKNOTURF

```
Create view customer_policy_details as Select cname,pname,duedate
from customer c join policyenrollment p  on c.cid=p.cid join policy pp on
p.pid=pp.pid;
```

```
Create view policy_details as select pid,pname from policy where
MinAmountPerMonth>=500 with check option;
```

WITH CHECK OPTION is to ensure that all UPDATE and INSERT satisfy the condition(s) in the view definition.

Rules on Views for DML

TEKNOTURF

You can perform DML Operations on a simple view.

You cannot remove a row if the view contains

- Group functions
- GROUP BY clause
- DISTINCT keyword

You cannot modify a row if the view contains

- Group functions
- GROUP BY clause
- DISTINCT keyword
- Column defined by expressions

You cannot add a row if the view contains

- Group functions
- GROUP BY clause
- DISTINCT keyword
- Column defined by expressions
- NOT NULL columns in the base tables not selected in the view.

Inline views

TEKNOTURF

An inline view is a subquery with an alias that can be used within an SQL statement.

A named subquery in the FROM clause of the main query is an example of an inline view.

Removing View

TEKNOTURF

Syntax

- Drop view viewname;

Example

- Drop view policy_details;

Top N Analysis

TEKNOTURF

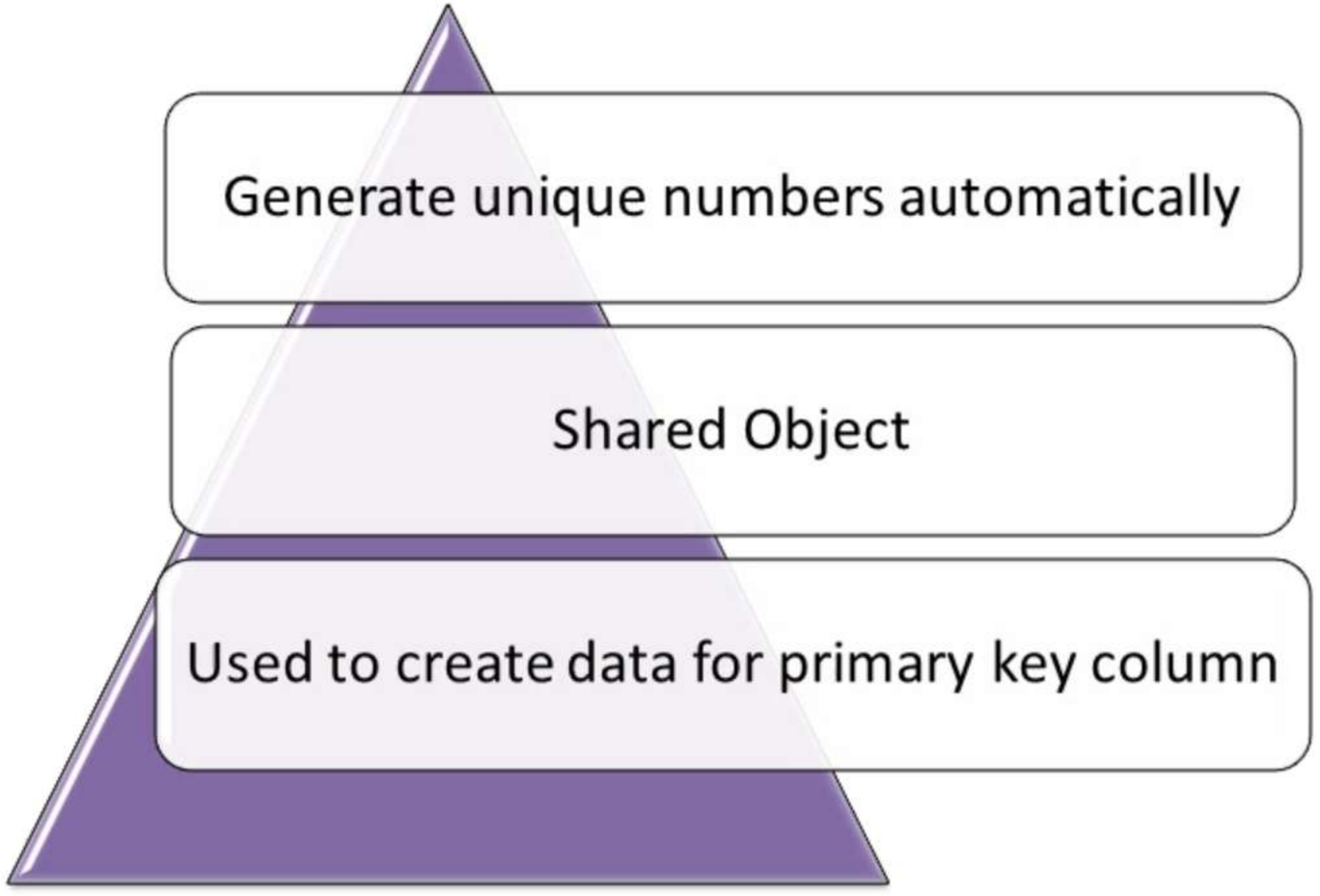
Query to display the first three maximum priced policy from the policy table.

```
SELECT ROWNUM as RANK, pname, MinAmontPerMonth
FROM (SELECT pname,MinAmountPerMonth FROM policy
ORDER BY MinAmountPerMonth DESC)WHERE ROWNUM <= 3;
```

Top-n queries retrieve the n largest or smallest values of a column.

Sequence

TEKNOTURF



Sequence

TEKNOTURF

Maxvalue: maximum generated by sequence

Minvalue: minimum sequence value

Cycle : specifies whether to continue generating sequence value after reaching maxvalue

Cache : specifies how many values the Oracle Server pre-allocates and keeps in memory

```
CREATE SEQUENCE seq_name
    [increment by n]
    [start with n]
    [maxvalue n | nomaxvalue]
    [minvalue n | nominvalue]
    [cycle | nocycle]
    [cache | nocache]
```

Sequence Psuedocolumns

TEKNOTURF

NEXTVAL returns the next available sequence value.

- It returns a unique value every time it is referenced, even for different users.

CURRVAL returns the current sequence value.

- NEXTVAL must be issued for that sequence before CURRVAL contains a value.

Sequence - Example

TEKNOTURF

create and use sequence

CREATING A SEQUENCE

```
CREATE SEQUENCE EMP_SEQ START WITH 1;
```

NEXTVAL

```
INSERT INTO EMP VALUES(EMP_SEQ.NEXTVAL,'MINI');
```

CURRVAL

```
SELECT EMP_SEQ.CURRVAL FROM DUAL;
```

Modifying Sequence

TEKNOTURF

Change the increment value, maximum value, minimum value, cycle option, or cache option.

Maxvalue should no be less than the current value

Start with cannot be changed

Future sequence number only will be affected

```
ALTER SEQUENCE emp_seq
INCREMENT BY 20
MAXVALUE 999999
NOCACHE
NOCYCLE;
```

USER_SEQUENCES

How to get the sequence information?

```
select min_value, max_value,
increment_by from user_sequences
where sequence_name='EMP_SEQ';
```

User_sequences – a data dictionary

How to drop the created sequence?

```
DROP SEQUENCE BOOK_SEQ;
```

Synonym

TEKNOTURF

What is a synonym?

Creating an alias Name

For what should a synonym be created?

Tables, views, sequence and other schema object.

How to create it?

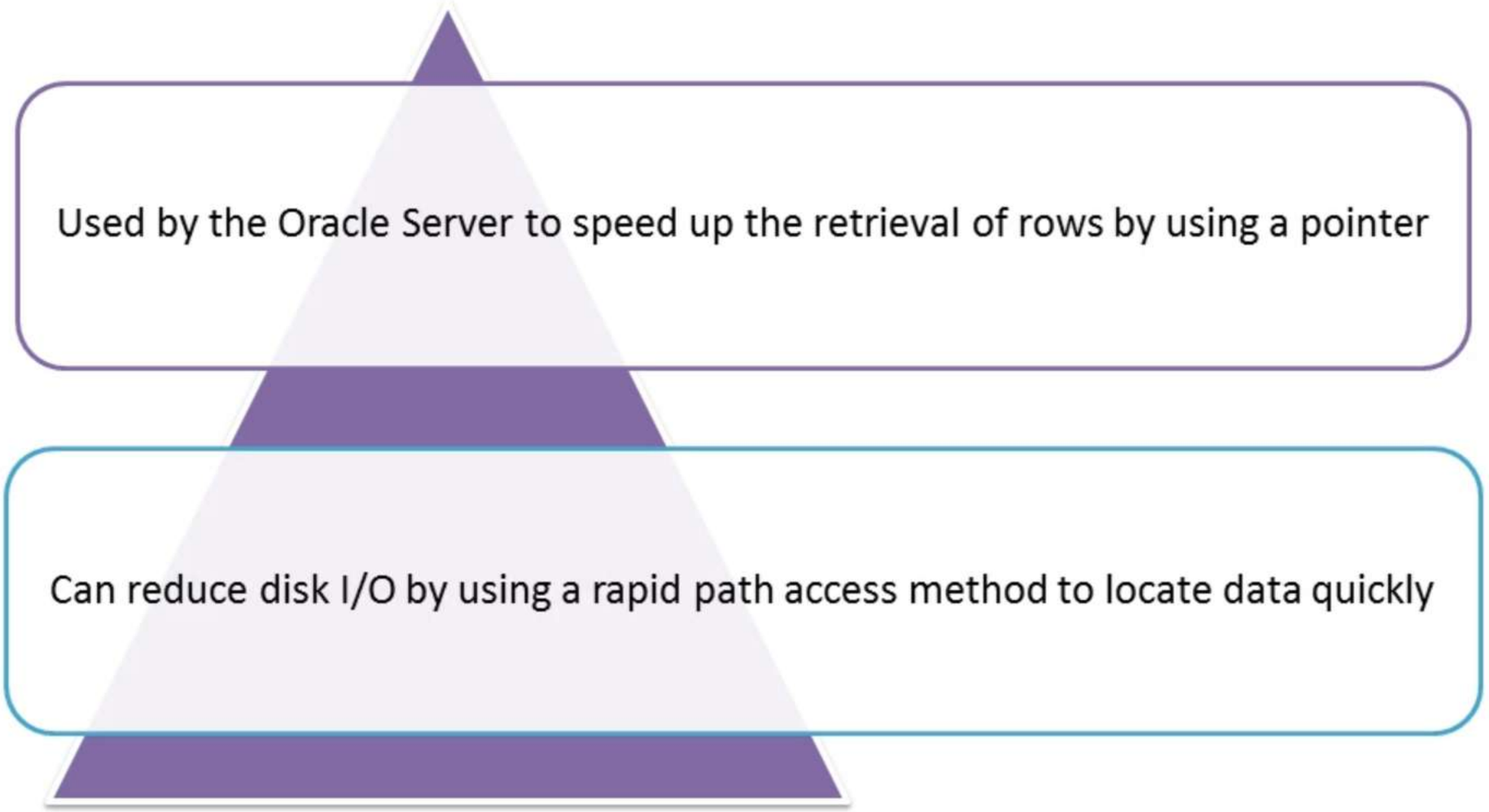
CREATE [PUBLIC] SYNONYM SYN_NAME FOR SCHEMA_OBJ

```
CREATE SYNONYM ENROLL FOR POLICYENROLLMENT;
```

```
DROP SYNONYM ENROLL
```

Index

TEKNOTURF



Index

TEKNOTURF

Create Index

- CREATE [UNIQUE] INDEX INDEX_NAME ON TABLE(COLUMN)
- CREATE INDEX POLICY_IND ON POLICY(PNAME);

You should create an index if:

- A column contains a wide range of values.
- A column contains a large number of null values.
- One or more columns are frequently used together in a WHERE clause or a JOIN condition
- The table is large.

Index Types

TEKNOTURF

Clustered index

- Creates an index which sorts the data file in the order of the index file
- Can have only one clustered index
- Automatically gets created for Primary key
- CREATE CLUSTERED INDEX POLICY_IND1 ON POLICY(PID);

Non-clustered index

- Can be created for all the columns
- More than 1 index can be created
- Data are not sorted according to the order of index file
- CREATE INDEX POLICY_IND2 ON POLICY(PNAME);

Index Types

TEKNOTURF

Unique index

- Makes the column for which the index is created to have a unique data
- Automatically created for columns with unique constraint
- CREATE UNIQUE INDEX CUSTOMER_IND1 ON CUSTOMER(EMAILID);

Function based index

- A function-based index is an index based on expressions.
- CREATE INDEX CUSTOMER_IND2 ON CUSTOMER(UPPER(CNAME));

Index

TEKNOTURF

Viewing index information

- Data dictionary used:
  - User_indexes
  - User_ind_columns

Dropping an Index

- DROP INDEX <INDEX_NAME>
- DROP INDEX POLICY_IND;

Summary

TEKNOTURF

- Grant
- Revoke
- View
- Inline view
- Top N analysis
- Sequence
- Synonym
- Index