Database Security

CT069-3-3

Structured Query Language (SQL)

Learning Outcomes

At the end of this topic, You should be able to

- Explain what is SQL and SQL artifacts such as database, schema, table, column, and view
- Write SQL **Data Definitions Language** queries to create, modify and remove tables, and views
- Design and maintain a database system with high integrity
- Write SQL DDL queries to add and remove columns of various data types
- Write SQL **Data Manipulation Language** queries to add, update and remove data
- Write basic and advance SQL Data Query L (Select) queries to retrieve data

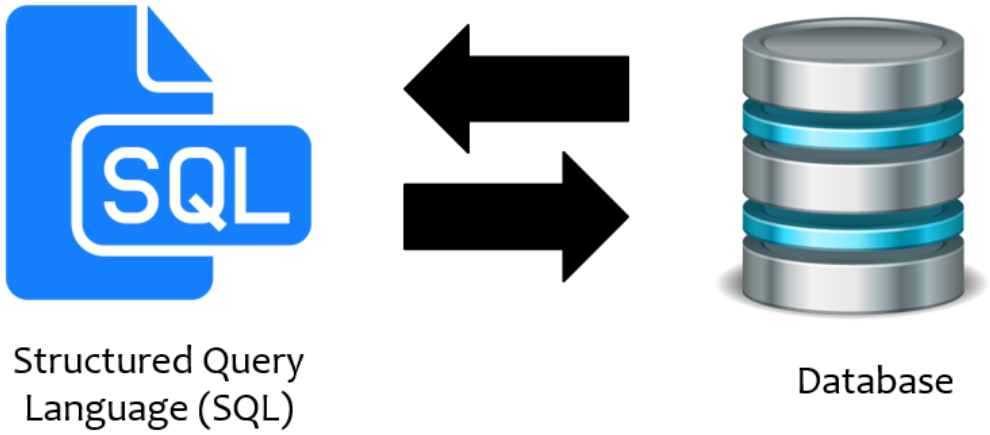
Key Terms you must be able to use

If you have mastered this topic, you should be able to use the following terms correctly :

- SQL
- SQL Table, View, Column, Constraint, Primary Key, Foreign Key
- DDL – Data Definition Language – create and modify structures
- DML – Data Manipulation Language - to manipulate data
- DQL – Data Query Language

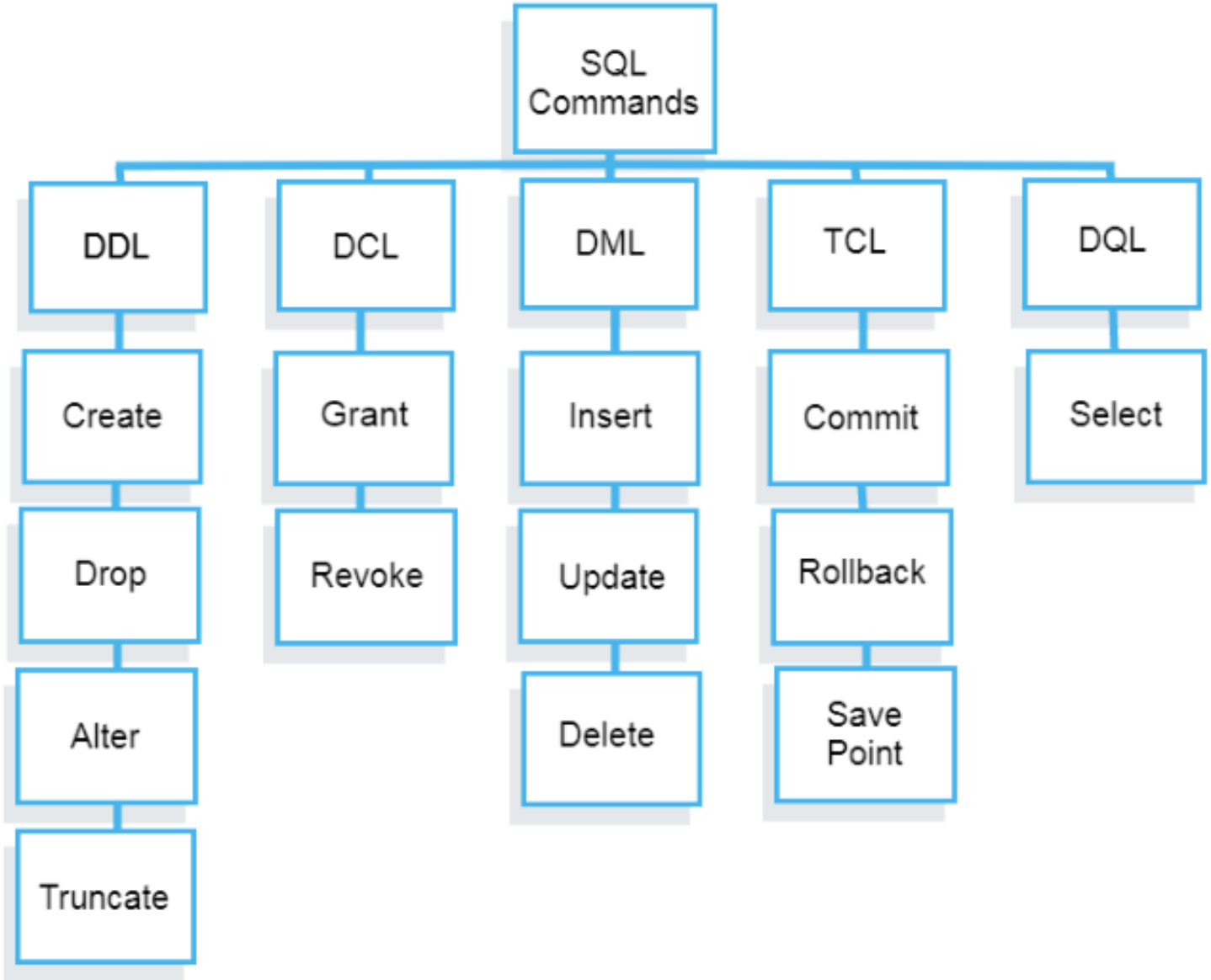
Structured Query Language (SQL)

- Structured query language **(SQL)** is the standard **programming language for interacting with** the Database Management System **(DBMS).**

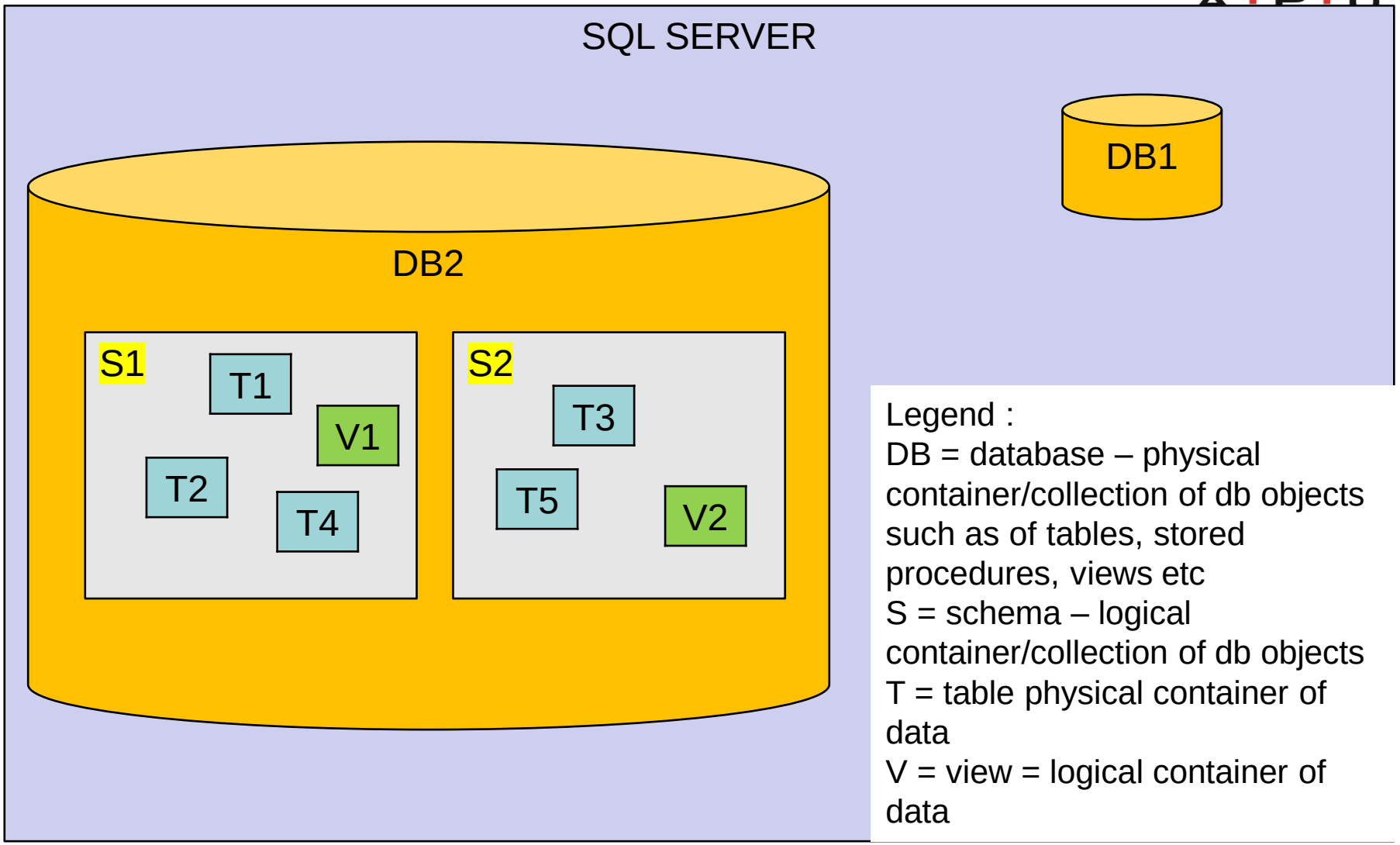


SQL

- SQL is a database language designed for the retrieval and management of data in a relational database.
- SQL is the standard language for database management.
- All the RDBMS systems such as MS SQL Server, MySQL and Oracle use SQL as their standard database language



DB Structure



Legend :
DB = database – physical container/collection of db objects such as of tables, stored procedures, views etc
S = schema – logical container/collection of db objects
T = table physical container of data
V = view = logical container of data

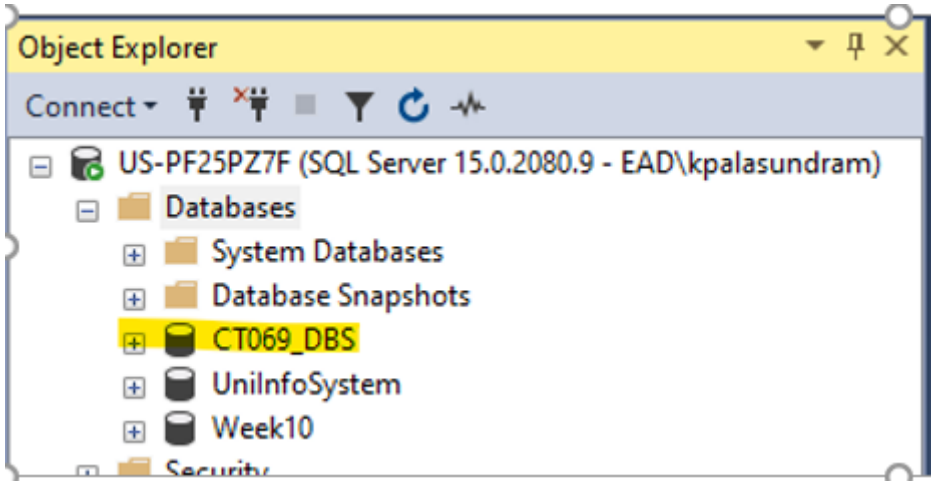
Database

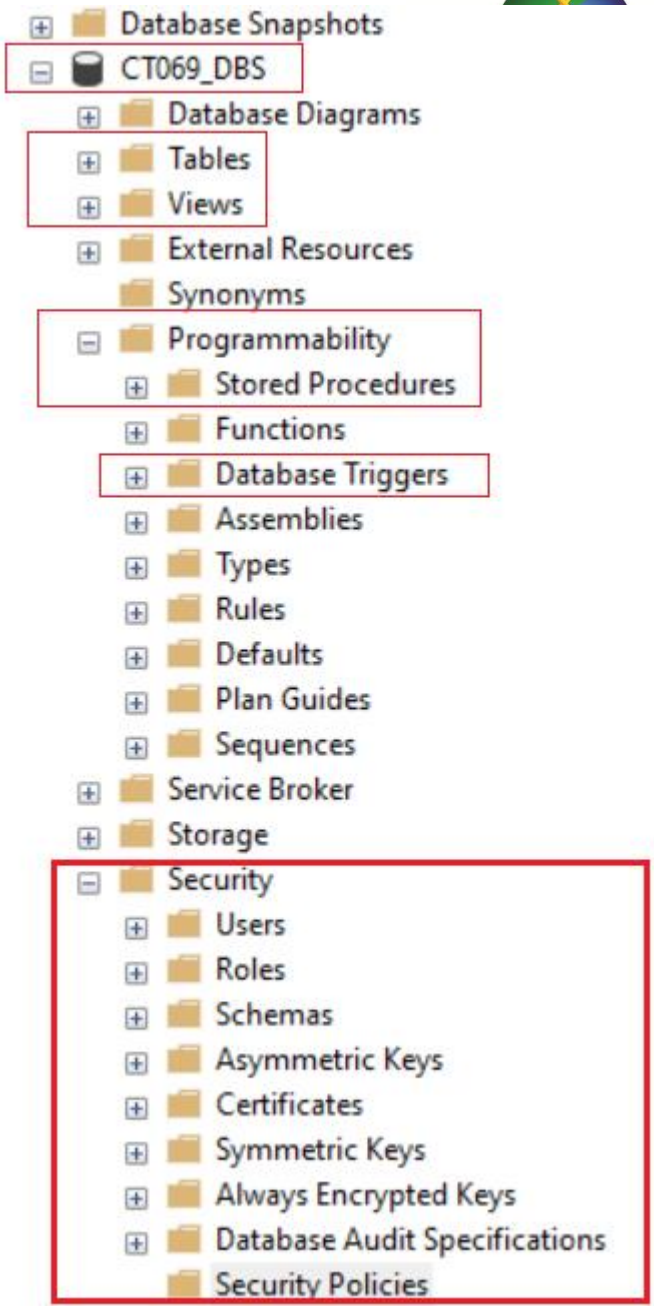
- An organized collection of **structured data** to make it easily accessible, manageable and update
- Many types of database – our focus is **relational database**
- Organizing data in database using DBMS provides:-
  - Reduced data redundancy.
  - Reduced updating errors and increased consistency.
  - Ensure data integrity from application programs.
  - Supported by SQL –structured query language
  - Better control on access and security

Creating a database

SQLQuery1.sql - US...kpalasundram (51))*

```
CREATE DATABASE CT069_DBS
```





Update or Remove Database

- `ALTER DATABASE [Week10] MODIFY NAME = [Test2]`
- `DROP DATABASE [Test2]`

Schema

- **Logical grouping of database objects**
- Can be assigned security permissions - an effective method for managing user access and privileges
- Schema always belong to a single database whereas a database can have single or multiple schemas

```
CREATE SCHEMA [student]

CREATE SCHEMA [staff]
```

```
DROP SCHEMA student
```

```
ALTER SCHEMA [student]
...
```

Table

- Purpose is to structure and store data required by the organization/solution
- Consists of rows and columns
- Independent and permanent data object
- It occupies space on the systems

Creating Table Structures

- Table and column names should best reflect the meaning
- Data Types
  - Column data type selection is usually dictated by nature of data and by intended use
  - Pay close attention to expected use of attributes for sorting and data retrieval purposes

MS-SQL Data Types – numerical

Data Type	Description	Lower limit	Upper limit	Memory
bigint	It stores whole numbers in the range given	−2^63 (−9,223,372, 036,854,775,808)	2^63−1 (−9,223,372, 036,854,775,807)	8 bytes
int	It stores whole numbers in the range given	−2^31 (−2,147, 483,648)	2^31−1 (−2,147, 483,647)	4 bytes
smallint	It stores whole numbers in the range given	−2^15 (−32,767)	2^15 (−32,768)	2 bytes
tinyint	It stores whole numbers in the range given	0	255	1 byte
bit	It can take 0, 1, or NULL values.	0	1	1 byte/8bit column
decimal	Used for scale and fixed precision numbers	−10^38+1	10^381−1	5 to 17 bytes
money	Used monetary data	−922,337, 203, 685,477.5808	+922,337, 203, 685,477.5807	8 bytes
smallmoney	Used monetary data	−214,478.3648	+214,478.3647	4 bytes

MS-SQL Data Types – date and time

Data Type	Description	Storage size	Accuracy	Lower Range	Upper Range
DateTime	Used for specifying a date and time from January 1, 1753 to December 31, 9999. It has an accuracy of 3.33 milliseconds.	8 bytes	Rounded to increments of .000, .003, .007	1753-01-01	9999-12-31
smalldatetime	Used for specifying a date and time from January 1, 0001 to December 31, 9999. It has an accuracy of 100 nanoseconds	4 bytes, fixed	1 minute	1900-01-01	2079-06-06
date	Used to store only date from January 1, 0001 to December 31, 9999	3 bytes, fixed	1 day	0001-01-01	9999-12-31
time	Used for storing only time only values with an accuracy of 100 nanoseconds.	5 bytes	100 nanoseconds	00:00:00.0000000	23:59:59.9999999

MS-SQL Data Types – string

Data Type	Description	Lower limit	Upper limit	Memory
char	It is a character string with a fixed width. It stores a maximum of 8,000 characters.	0 chars	8000 chars	n bytes
varchar	This is a character string with variable width	0 chars	8000 chars	n bytes + 2 bytes
varchar (max)	This is a character string with a variable width. It stores a maximum of 1,073,741,824 characters.	0 chars	2^31 chars	n bytes + 2 bytes
text	This is a character string with a variable width. It stores a maximum 2GB of text data.	0 chars	2,147,483,647 chars	n bytes + 4 bytes

SQL Constraints

- SQL constraints are used to specify rules for the data in a table. Constraints 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 table.**
- Common type of Constraints
  - NOT NULL constraint : Ensures that column does not accept nulls
  - UNIQUE constraint : Ensures that all values in column are unique
  - PRIMARY KEY Constraint : ***Note: Primary key implements both a NOT NULL and UNIQUE constraint***
  - FOREIGN KEY Constraint.
  - DEFAULT constraint : Assigns value to attribute when a new row is added to table if the value is not provided by the user
  - CHECK constraint :Validates data when attribute value is entered

Table DDL Statements

- To create a new table in the database

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

Note: Default schema is dbo

- To remove or delete the whole table
  - DROP TABLE *schema.table_name* ;

  Note: DROP table will also remove all the data permanently

- To modify the structure of an existing table
  - ALTER TABLE *schema.table_name*
    <perform actions like adding, dropping, modifying a column in the table etc>

Data Dictionary

- Table name: RegistrationStatus

Column Name	Data Type (Length)	Default Value (if any)	Note
StatusID	Varchar(1)		Primary Key
StatusDescription	Varchar(10)		Must be unique

- Table name: Student

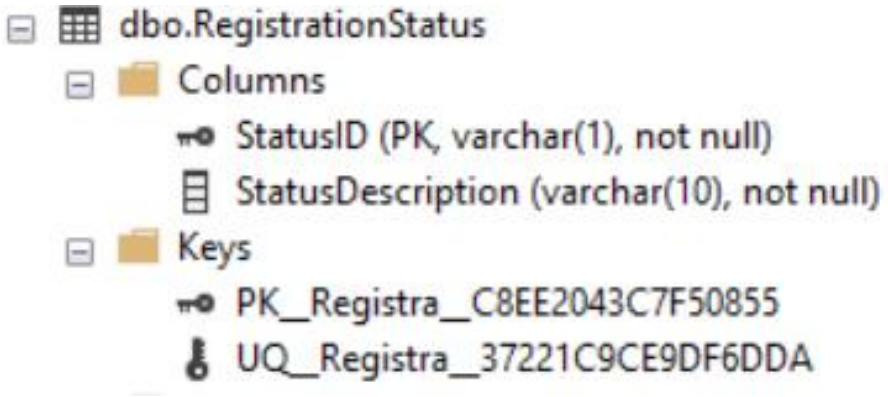
Column Name	Data Type (Length)	Default Value (if any)	Note
StudentID	Varchar(5)		Primary Key
Name	Varchar(100)		Cannot be null
RegistrationDate	Date		
CurrentYear	Int		Must be positive integer
Status	Varchar(1)	1	Foreign key to RegistrationStatus(StatusID)

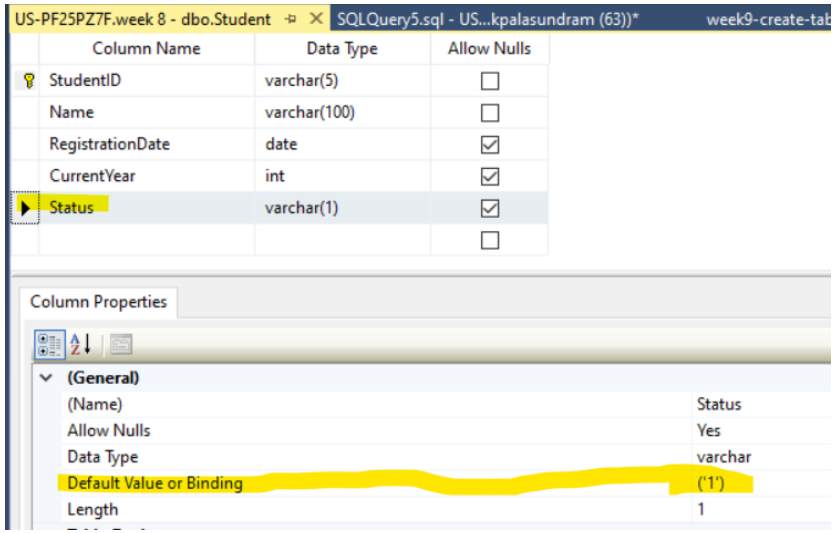
Create Table - Samples

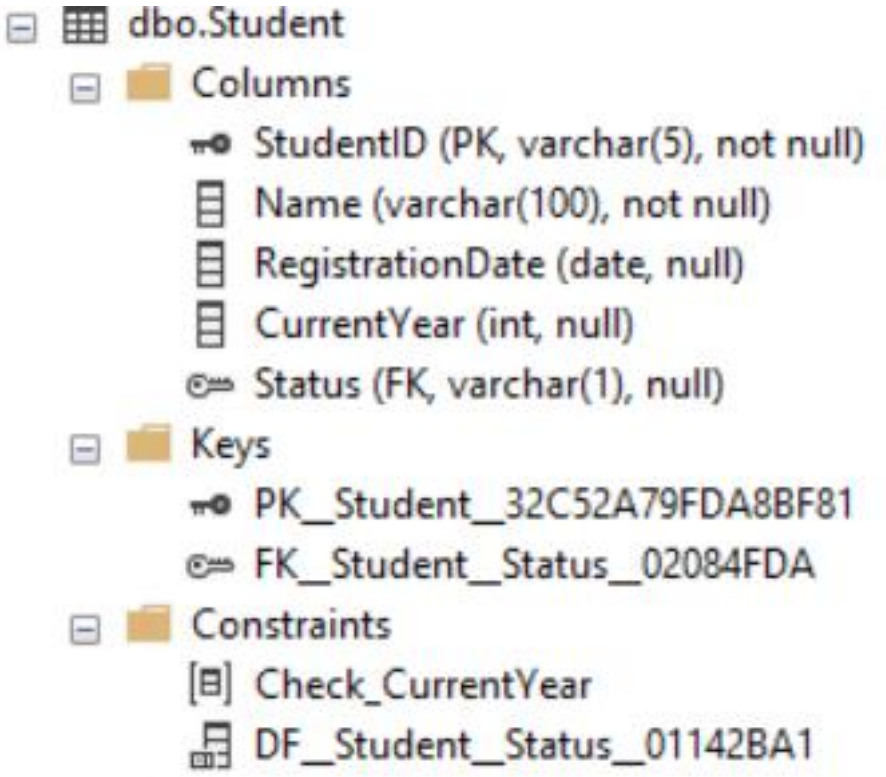
A · P · U
ASIA PACIFIC UNIVERSITY
OF TECHNOLOGY & INNOVATION

```
Create Table RegistrationStatus
(
  StatusID varchar(1) primary key,
  StatusDescription varchar(10) unique
)

Create Table [Student]
(
  StudentID varchar(5) primary key (StudentID),
  Name varchar(100) not null,
  RegistrationDate date,
  CurrentYear int ,
  CONSTRAINT Check_CurrentYear CHECK (CurrentYear > 0),
  [Status] varchar(1) default '1',
  FOREIGN KEY ([Status]) REFERENCES RegistrationStatus(StatusID)
)
```

The tables in MS-SQL







Ensuring Data Integrity

A·P·U
ASIA PACIFIC UNIVERSITY
OF TECHNOLOGY & INNOVATION

- Consider four types of integrity constraints:
  - Required data.
  - Domain constraints.
  - Entity integrity.
  - Referential integrity

Integrity Constraint: Required Data

- Ensuring any required or mandatory data are always present in the database
- Enforced using the **NOT NULL constraint**
- For example, let's say the Book Title and Price is the required data for the Book Table.
- We need to define the columns to have NOT NULL constraint

Column name	SQL Data Type
Book ID	Integer
Category	Varchar (2)
Title	**Varchar (200)**
Price	**Decimal (4,2)**
Published Date	Date
Publisher ID	Varchar (3)
Quantity In Stock	SmallInt
Authors	Varchar (MAX)

Integrity Constraint: Required Data

```
CREATE TABLE Book
(
        Bookid SMALLINT IDENTITY PRIMARY KEY,
        BookTitle varchar(200) not null,
        Price Decimal (4,2) not null
)
```

Bookid	Book Title	Price
1	Rainbow	30.50
2	Coffee Break	25.50

Integrity Constraint: Required Data

```
INSERT into Book values('Database',NULL)
```

```
Messages
Msg 515, Level 16, State 2, Line 14
Cannot insert the value NULL into column 'Price', table 'CT069_DBS.dbo.Book';
column does not allow nulls. INSERT fails.
The statement has been terminated.
```

```
UPDATE Book SET Price = NULL Where BookTitle = 'Rainbow'
```

```
Messages
Msg 515, Level 16, State 2, Line 13
Cannot insert the value NULL into column 'Price', table 'CT069_DBS.dbo.Book';
column does not allow nulls. UPDATE fails.
The statement has been terminated.
```

Integrity Constraint: Domain Constraint

- Domain constraint refers to valid data type, values and size that the column can store
- This is implemented automatically by DBMS based on **column definition** and the **CHECK** constraint

```
Create Table Student
(
  StudentID varchar(5) primary key (StudentID),
  Name varchar(100) not null,
  Gender char(1) not null ,
  Constraint Check_Gender CHECK (Gender in ('F','M')),
  RegistrationDate date not null,
  CurrentYear int ,
  CONSTRAINT Check_CurrentYear CHECK (CurrentYear > 0),
  [Status] varchar(1) default '1',
  FOREIGN KEY ([Status]) REFERENCES RegistrationStatus(StatusID)
)
```

Domain Constraint Examples

- "Registration Date" column can store date values only
- "Current Year" can store integer values only and they must be more than 0
- "Status" column can store 1 character only
- 'Gender' column can store 1 character only and it must be either 'F' or 'M' only

```
INSERT INTO [dbo].[Student]
           ([StudentID]
           ,[Name]
           ,[Gender]
           ,[RegistrationDate]
           ,[CurrentYear]
           ,[Status])
     VALUES
           ('S1000','John','M','not date', 1, '1')
GO
```

```
107 %
Messages
Msg 241, Level 16, State 1, Line 4
Conversion failed when converting date and/or time from character string

Completion time: 2022-05-20T15:30:57.5821510+08:00
```

```
INSERT INTO [dbo].[Student]
           ([StudentID]
           ,[Name]
           ,[Gender]
           ,[RegistrationDate]
           ,[CurrentYear]
           ,[Status])
     VALUES
           ('S1000','John','Z',getdate(), 1, '1')
GO
```

```
7 %
Messages
Msg 547, Level 16, State 0, Line 4
The INSERT statement conflicted with the CHECK constraint "Check_Gender".
The conflict occurred in database "CT069_DBS", table "dbo.Student", column 'Gender'.
The statement has been terminated.

Completion time: 2022-05-20T15:32:09.8544964+08:00
```

Entity Integrity

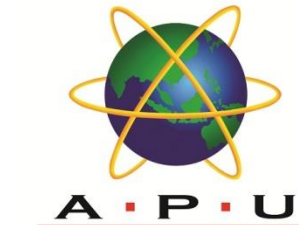
- Entity integrity is to ensure that each row in a table represents a single instance of the entity type modelled by the table.
- To ensure *entity integrity*, it is required that every table have a key which is unique and non-null.
- **PRIMARY KEY constraint implements both unique and non-null implicitly.**
- This is because null values for the primary key mean we cannot identify some rows.
- We can only have one PRIMARY KEY clause per table.

Entity Integrity

- Can still ensure uniqueness for alternate keys or other columns by using the UNIQUE keyword

```
Create Table RegistrationStatus
(
  StatusID varchar(1) primary key (StatusID),
  StatusDescription varchar(10) unique
)
```

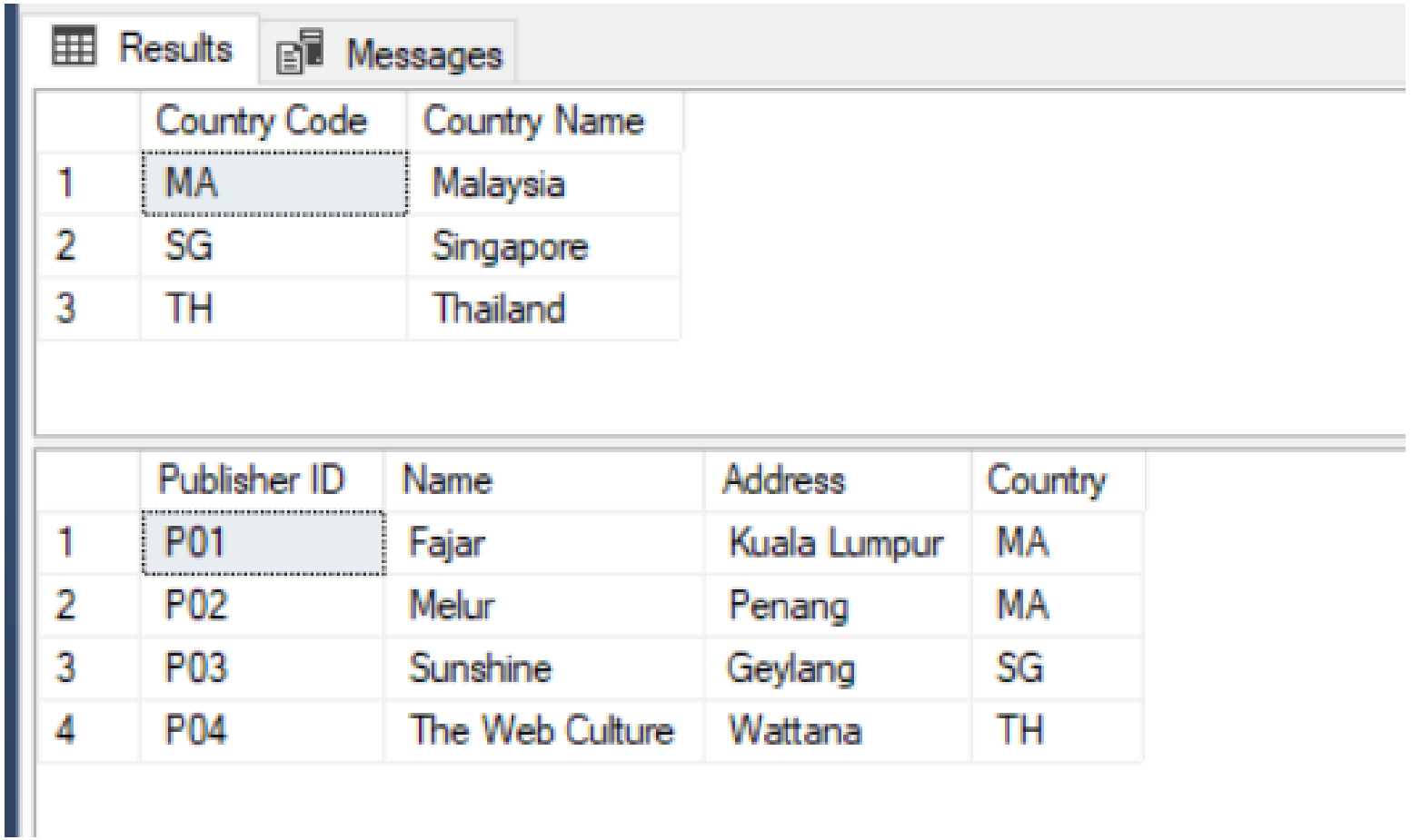
Referential Integrity



- Referential integrity means that, if FK contains a value, that value must refer to existing row in parent table.
- Foreign Key (FK) is column or set of columns that links each row in child table containing foreign FK to row of parent table containing matching Primary Key (PK).

```
create table [BookStore].[Country]
(
[Country Code] varchar(2) primary key,
[Country Name] varchar(100)
)

create table [BookStore].[Publisher]
(
[Publisher ID] varchar(3) primary key,
[Name] varchar(200),
[Address] varchar(200),
[Country] varchar(2) references [BookStore].[Country]([Country Code])
)
```

Sample Values



Results Messages

	Country Code	Country Name
1	MA	Malaysia
2	SG	Singapore
3	TH	Thailand

	Publisher ID	Name	Address	Country
1	P01	Fajar	Kuala Lumpur	MA
2	P02	Melur	Penang	MA
3	P03	Sunshine	Geylang	SG
4	P04	The Web Culture	Wattana	TH

Referential Integrity

- INSERT operation will be rejected if it attempts to **create FK value in child table** without matching candidate key value in parent

```
--INSERT a new Publisher with non existing country code
INSERT INTO BookStore.Publisher
VALUES ('P05','Anything','Kuala Lumpur','JP')
```

7 %

Messages

```
Msg 547, Level 16, State 0, Line 13
The INSERT statement conflicted with the FOREIGN KEY constraint "FK__Publisher__Count__2B3F6F97".
 The conflict occurred in database "APU Database", table "BookStore.Country", column 'Country Code'.
The statement has been terminated.

Completion time: 2022-05-24T14:19:24.8460012+08:00
```

Referential Integrity

- UPDATE operation will be rejected it attempts to **change the FK value in child table** without matching candidate key value in parent is rejected.

```
UPDATE [BookStore].[Publisher]
SET [Country] = 'US'
WHERE [Name] = 'Fajar'
```

107 %

Messages

```
Msg 547, Level 16, State 0, Line 16
The UPDATE statement conflicted with the FOREIGN KEY constraint "FK__Publisher__Count__2B3F6F97".
The conflict occurred in database "APU Database", table "BookStore.Country", column 'Country Code'.
The statement has been terminated.

Completion time: 2022-05-24T14:32:33.3377127+08:00
```

Referential Integrity

- DELETE operation will be rejected it attempts to **delete a row on parent table** which has an FK referencing to it

```
DELETE FROM [BookStore].[Country]
WHERE [Country Name]='Malaysia'
```

```
Messages
Msg 547, Level 16, State 0, Line 16
The DELETE statement conflicted with the REFERENCE constraint "FK__Publisher__Count__2B3F6F97".
The conflict occurred in database "APU Database", table "BookStore.Publisher", column 'Country'.
The statement has been terminated.
```

Referential Integrity

- Action taken that attempts to update/delete a candidate key value in parent table with matching rows in child is dependent on referential action specified using ON UPDATE and ON DELETE subclauses:
  - CASCADE
  - SET NULL
  - SET DEFAULT
  - NO ACTION - Reject delete from parent. Default

Usage - Cascade

- ON DELETE CASCADE
  - Delete all rows in parent and matching records in child
- ON UPDATE CASCADE
  - Update all rows in child and parent with the new values

```
create table [BookStore].[Country_2]
(
[Country Code] varchar(2) primary key,
[Country Name] varchar(100)
)

create table [BookStore].[Publisher_2]
(
[Publisher ID] varchar(3) primary key,
[Name] varchar(200),
[Address] varchar(200),
[Country] varchar(2) references [BookStore].[Country_2]([Country Code])
ON DELETE CASCADE
ON UPDATE CASCADE
)
```

Usage – Set Null

- ON DELETE SET NULL
  - Delete all rows in parent and updates the FK in child to NULL
- ON UPDATE SET NULL
  - Update all rows in parent with the new values and updates the FK in child to NULL

```
create table [BookStore].[Publisher_2]
(
[Publisher ID] varchar(3) primary key,
[Name] varchar(200),
[Address] varchar(200),
[Country] varchar(2) Default 'DF'
References [BookStore].[Country_2]([Country Code])
ON DELETE SET NULL
ON UPDATE SET NULL
)
```

Usage – Set Default

- ON DELETE SET DEFAULT
  - Delete row from parent and set each component of FK in child to specified default.
- ON UPDATE SET DEFAULT
  - Update all rows in parent with the new values and updates the FK in child to specified default.

```
create table [BookStore].[Publisher_2]
(
[Publisher ID] varchar(3) primary key,
[Name] varchar(200),
[Address] varchar(200),
[Country] varchar(2) Default 'DF'
References [BookStore].[Country_2]([Country Code])
ON DELETE SET DEFAULT
ON UPDATE SET DEFAULT
)
```

Auto Increment

- Auto-increment feature allows a unique number to be generated automatically when a new record is inserted into a table.
- Often this is the primary key field that we would like to be populated automatically every time a new record is inserted.

Data Dictionary

- Table name: Order

Column Name	Data Type (Length)	Default Value (if any)	Note
OrderID	Int		Primary Key, Automatically generated incremental integer
Amount	Decimal (10,2)		
OrderDate	Date	Today's date	

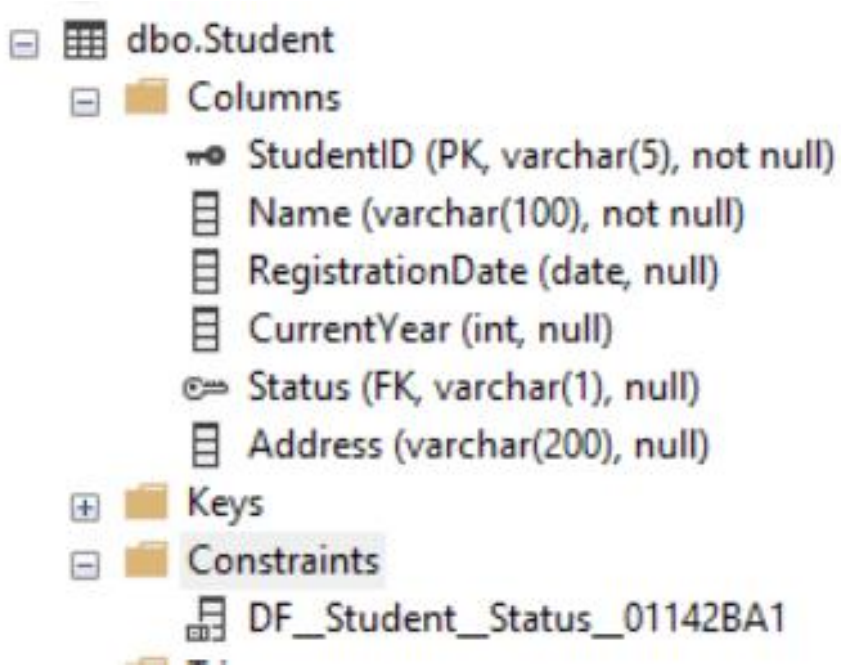
Auto Increment - Sample


```
Create Table [Order]
(
  OrderID int IDENTITY(100,1) primary key,
  Amount decimal(10,2),
  OrderDate date Default getdate(),
  OrderDateTime datetime Default getdate()
)
```

```
alter table [order]
add [OrderDateTime] datetime default getdate()
```

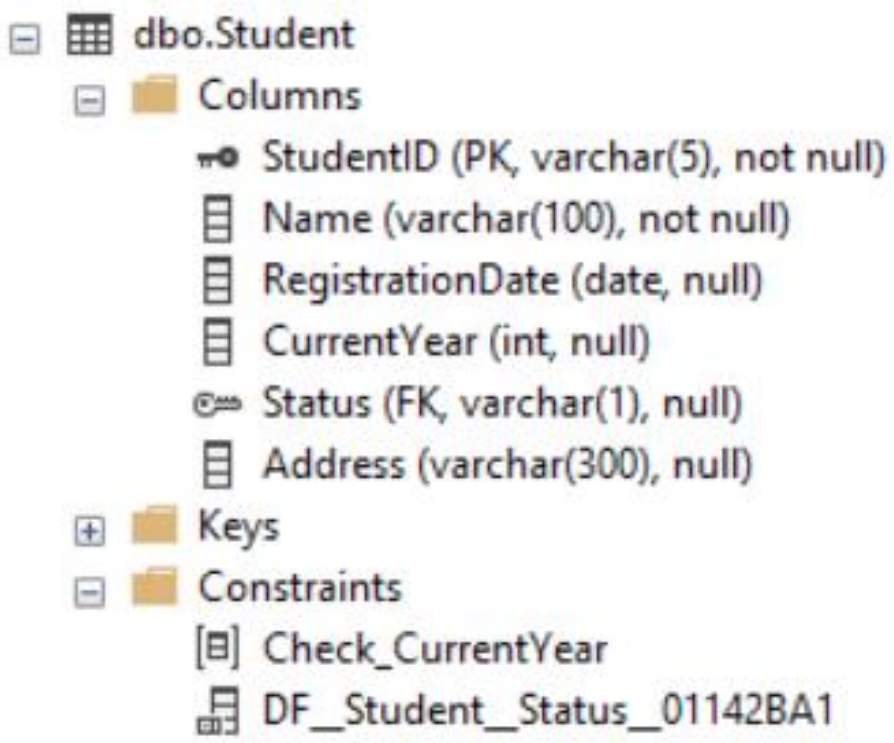
Alter Table - Sample

```
Alter Table Student Add [Address] varchar(200);

Alter Table Student Drop Constraint Check_CurrentYear;
```



Alter Table - Sample

```
Alter Table Student Alter Column [Address] varchar(300);

Alter Table Student Add Constraint Check_CurrentYear CHECK
(CurrentYear > 0);
```



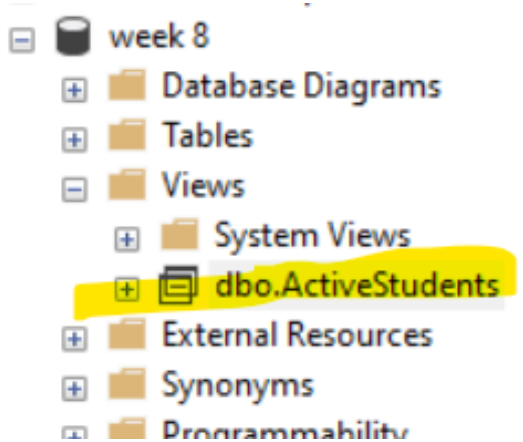
SQL View

- A virtual/logical table formed as a result of a query and used to view or manipulate parts of the table.
- Its content is based on base table.
- May contain data from one or more tables.
- It does not occupy space on our systems.
- **Purposes are to**
  - **Simplify writing frequently used queries especially if the results involves data from multiple tables**
  - **Improved query speed because RDBMS will optimize your views**
  - **Hide the actual table and column names for security reasons**
  - **Show column names in a more presentable format**

Create View - Sample

```
CREATE OR ALTER View [dbo].[ActiveStudents] As
Select name as [Student Name]
From Student
Where Status = 2
```

- Hides the table and column names and the status id value



View Usage - Sample

- `select * from student where Status = 2`
- `select * from ActiveStudents`

	StudentID	Name	RegistrationDate	CurrentYear	Status	Address
1	102	Sam	2022-01-01	1	2	NULL
2	106	Aaron	2022-01-01	1	2	NULL
3	110	Frazer	2022-01-01	1	2	NULL

	Student Name
1	Sam
2	Aaron
3	Frazer

Create View - Sample

```
CREATE OR ALTER View [dbo].[NonActiveStudents]
as
select student.name as [Student Name],
registrationstatus.StatusDescription as [Registration
Status]
from student , RegistrationStatus
where student.Status = RegistrationStatus.StatusID
and not RegistrationStatus.StatusDescription = 'Active'
```

- Combines 2 tables and present them as just 1 view (virtual table)

Create View - Sample

```
SELECT * FROM [NonActiveStudents]
```

9 %

Results | Messages

	Student Name	Registration Status
1	Sam	Completed
2	Ram	OnLeave
3	Aaron	Completed
4	Bao	OnLeave
5	Damion	OnLeave
6	Eeline	Completed
7	Frazer	Completed

DROP

- Drop Table <table name>
- Drop View <view name>

INSERT statements

- To add data into SQL tables, we need to use INSERT command
- There are various methods to write an insert statement
- It depends on the table structure / constraints

Insert – Method 1

Column Name	Data Type	Allow Nulls
[course id]	varchar(10)	☐
[course title]	varchar(100)	☑
[course unit]	smallint	☑

- Provide all the column names and corresponding values
- This is the most proper and safest statement

```
insert into course
([course id], [course title],[course unit])
values ('100','Biology',3)
```

Insert – Method 1

```
insert into course ( [course id], [course title],[course
description],[course unit])
values ('100','Biology','this is an introductory course biology for
first year students',3)

insert into course ( [course id], [course unit], [course
description],[course title])
values ('200',4,'this is an advance course for final year
students','Advance Chemistry')
```

course id	course title	course description	course unit
100	Biology	this is an introductory course biology for first y...	3
200	Advance Chemistry	this is an advance course for final year students	4

Insert – Method 2

A·P·U
ASIA PACIFIC UNIVERSITY
OF TECHNOLOGY & INNOVATION

- Provide only values based on the order of which the table columns was created
- This may cause issues if data was provided in wrong order

```
insert into course
values ('300','Maths','Basic mathematics for first year
student',2)

insert into course
values ('400','this is an advance course for final year
students','Advance Physics',4)
```

Method 2

Note that the wrong data entry in the last row

course id	course title	course description	course unit
100	Biology	this is an introductory course biology for first y...	3
200	Advance Chemistry	this is an advance course for final year students	4
300	Maths	Basic mathematics for first year student	2
400	this is an advance course for final year students	Advance Physics	4

Method 3
- Insert for Identity Column

- Identity column name **must be left out** from the insert statement when inserting into a table with identity column

```
create table course_v2
([course id] int identity ( 100, 100) primary key,
 [course title] varchar(100),
 [course description] varchar (1000),
 [course unit] smallint
)

insert into course_v2 (  [course title],[course description],[course unit])
values ('Biology','this is an introductory course biology for first year
students',3)

insert into course_v2 ( [course unit], [course description],[course title])
values (4,'this is an advance course for final year students','Advance
Chemistry')
```

Method 4 – Insert – Default Column

- If Default column name is left out from the insert statement, then SQL will insert the default value
- If Default column name and value is provided in the insert statement, then SQL will insert the value that is provided

Method 4 - samples

ASIA PACIFIC UNIVERSITY OF TECHNOLOGY & INNOVATION

```
create table course_v3
([course id] int identity ( 100, 100) primary key,
 [course title] varchar(100),
 [course description] varchar (1000),
 [course unit] smallint default 2
)
insert into course_v3 (  [course title],[course description])
values ('Biology','this is an introductory course biology for first year students')

insert into course_v3 ( [course unit], [course description],[course title])
values (4,'this is an advance course for final year students','Advance Chemistry')
```

course id	course title	course description	course unit
100	Biology	this is an introductory course biology for first y...	2
200	Advance Chemistry	this is an advance course for final year students	4

UPDATE statements

- SQL Update is used to modify the values of an existing record in the database

- General syntax is

UPDATE <table name>
SET column1 name = new value,
column2 name = new value
WHERE <condition>

DELETE statements

- SQL Delete is used to remove 1 or more existing records from the table

- General syntax is

```
DELETE FROM <table name>
WHERE <condition>
```

Question

- What is the difference between DROP and DELETE commands ?

SELECT

Common syntaxes

<u>To select all the columns and rows</u>

- SELECT * FROM table_name;

<u>To select all rows but filter in certain columns</u>

- SELECT column1, column2, ... FROM table_name;

<u>To filter certain columns and rows</u>

- SELECT column1, column2, ... FROM table_name WHERE condition;

Sample Data

Employee

EmpID	EmpName	EmailID
100	Nathan	nathan@abc.com
110	Ismail	ismail@abc.com
120	Jason	jason@abc.com
130	Sharon	sharon@abc.com
140	Tyson	tyson@abc.com

Client

ClientID	ClientName
100	Ramli
110	Au
120	Khatijah
130	Joe
140	Mohd
150	Jason
160	Nathan

AssignedEmployees

EmpID	ProjectID
100	KL100
120	KL100
140	SEL200
130	KL200
100	KL200

Project

ProjectID	ProjectMgr	ClientID	ProjectName	Location	ProjectStartDate	Value
JH200	NULL	150	Causeway Lighting	Johor	2022-05-25	250000
JH400	NULL	150	Community Housing	Johor	2021-10-14	3000000
KL100	110	130	Putra Jaya Raya	Kuala Lumpur	2022-03-10	12000000
KL200	100	150	KL Inner City Road Expansion	Kuala Lumpur	2022-01-20	4500000
PG300	130	110	Jawi River Bridge	Penang	2022-05-05	350000
SEL200	120	100	Kids Hostel	Selangor	2021-12-15	1500000

Select Query Samples

```
select * from Employee
```

Results | Messages

EmpID	EmpName	EmailID
100	Nathan	nathan@abc.com
110	Ismail	ismail@abc.com
120	Jason	jason@abc.com
130	Sharon	sharon@abc.com
140	Tyson	tyson@abc.com

```
select EmpName, EmpID from Employee
```

Results | Messages

EmpName	EmpID
Nathan	100
Ismail	110
Jason	120
Sharon	130
Tyson	140

```
select * from Project where ProjectMgr IS NULL
```

Results | Messages

ProjectID	ProjectMgr	ClientID	ProjectName	Location	ProjectStartDate	Value
JH200	NULL	150	Causeway Lighting	Johor	2022-05-25	250000
JH400	NULL	150	Community Housing	Johor	2021-10-14	3000000

Select Query Samples

```
select * from Project where value > 3000000
```

Results | Messages

ProjectID	ProjectMgr	ClientID	ProjectName	Location	ProjectStartDate	Value
KL100	110	130	Putra Jaya Raya	Kuala Lumpur	2022-03-10	12000000
KL200	100	150	KL Inner City Road Expansion	Kuala Lumpur	2022-01-20	4500000

```
select GETDATE() as 'Today''s Date'
select * from Project where ProjectStartDate > GETDATE()
```

Results | Messages

Today's Date
2022-04-27 13:45:21.393

ProjectID	ProjectMgr	ClientID	ProjectName	Location	ProjectStartDate	Value
JH200	NULL	150	Causeway Lighting	Johor	2022-05-25	250000
PG300	130	110	Jawi River Bridge	Penang	2022-05-05	350000

Additional Commands to filter (used with WHERE clause)

- =, >, <, <>, !=, >=, <= - for numerical and date comparisons and string values
- LIKE – for partial comparison
- IN – for list comparison
- NOT - negation
- BETWEEN – between 2 values comparison
- AND – fulfill all conditions
- OR – fulfill one of the conditions

More Samples

```
select * from Project
where ProjectName like '%road%'
```

Results | Messages

ProjectID	ProjectMgr	ClientID	ProjectName	Location	ProjectStartDate	Value
KL200	100	150	KL Inner City Road Expansion	Kuala Lumpur	2022-01-20	4500000

```
select * from Project  where projectname not like
'%road%'
```

Results | Messages

	ProjectID	ProjectMgr	ClientID	ProjectName	Location	ProjectStartDate	Value
1	JH200	NULL	150	Causeway Lighting	Johor	2022-05-25	250000
2	JH400	NULL	150	Community Housing	Johor	2021-10-14	3000000
3	KL100	110	130	Putra Jaya Raya	Kuala Lumpur	2022-03-10	12000000
4	PG300	130	110	Jawi River Bridge	Penang	2022-05-05	350000
5	SEL200	120	100	Kids Hostel	Selangor	2021-12-15	1500000

More Samples

```
select * from Project  where location in ('Kuala Lumpur','Selangor')

select * from Project  where location = 'Kuala Lumpur'
or location = 'Selangor'
```

Results | Messages

ProjectID	ProjectMgr	ClientID	ProjectName	Location	ProjectStartDate	Value
SEL200	120	100	Kids Hostel	Selangor	2021-12-15	1500000
KL200	100	150	KL Inner City Road Expansion	Kuala Lumpur	2022-01-20	4500000
KL100	110	130	Putra Jaya Raya	Kuala Lumpur	2022-03-10	12000000

ProjectID	ProjectMgr	ClientID	ProjectName	Location	ProjectStartDate	Value
SEL200	120	100	Kids Hostel	Selangor	2021-12-15	1500000
KL200	100	150	KL Inner City Road Expansion	Kuala Lumpur	2022-01-20	4500000
KL100	110	130	Putra Jaya Raya	Kuala Lumpur	2022-03-10	12000000

```
select * from Project
where clientid=150 and location = 'Johor'
```

Results | Messages

ProjectID	ProjectMgr	ClientID	ProjectName	Location	ProjectStartDate	Value
JH200	NULL	150	Causeway Lighting	Johor	2022-05-25	250000
JH400	NULL	150	Community Housing	Johor	2021-10-14	3000000

More Samples

```
select * from Project  where value between 1000000 and 2000000

select * from Project  where value >= 1000000 and value <= 2000000
```

Results | Messages

ProjectID	ProjectMgr	ClientID	ProjectName	Location	ProjectStartDate	Value
SEL200	120	100	Kids Hostel	Selangor	2021-12-15	1500000

ProjectID	ProjectMgr	ClientID	ProjectName	Location	ProjectStartDate	Value
SEL200	120	100	Kids Hostel	Selangor	2021-12-15	1500000

Additional Commands that can be used in conjunction with Select

- Distinct – retrieve unique values
- Top n – get the first n rows
- Order By – arrange in ascending or descending
- Group By – summarize data

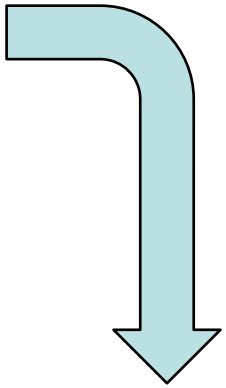
Distinct

```
select EmpID from AssignedEmployees
```

Results | Messages

EmpID
100
120
140
130
100

We may have repeating values



Distinct – shows unique values of EmpID by removing duplicates

```
select distinct EmpID from AssignedEmployees
```

Results | Messages

EmpID
100
120
130
140

TOP examples

- Get the project with the highest project value

```
select top 1 * from project
order by value desc
```

Results | Messages

ProjectID	ProjectMgr	ClientID	ProjectName	Location	ProjectStartDate	Value
KL100	110	130	Putra Jaya Raya	Kuala Lumpur	2022-03-10	12000000

TOP examples

- Identify the top 2 clients by project value

```
select top 2 clientid , sum(value) from project
group by clientid
order by sum(value) desc
```

Results | Messages

clientid	(No column name)
130	12000000
150	7750000

Order By


```
select * from employee order by empname asc
```

Results | Messages

EmpID	EmpName	EmailID
110	Ismail	ismail@abc.com
120	Jason	jason@abc.com
100	Nathan	nathan@abc.com
130	Sharon	sharon@abc.com
140	Tyson	tyson@abc.com

```
select * from project order by value desc
```

Results | Messages

ProjectID	ProjectMgr	ClientID	ProjectName	Location	ProjectStartDate	Value
KL100	110	130	Putra Jaya Raya	Kuala Lumpur	2022-03-10	12000000
KL200	100	150	KL Inner City Road Expansion	Kuala Lumpur	2022-01-20	4500000
JH400	NULL	150	Community Housing	Johor	2021-10-14	3000000
SEL200	120	100	Kids Hostel	Selangor	2021-12-15	1500000
PG300	130	110	Jawi River Bridge	Penang	2022-05-05	350000
JH200	NULL	150	Causeway Lighting	Johor	2022-05-25	250000

```
select * from project order by ProjectStartDate asc
```

Results | Messages

ProjectID	ProjectMgr	ClientID	ProjectName	Location	ProjectStartDate	Value
JH400	NULL	150	Community Housing	Johor	2021-10-14	3000000
SEL200	120	100	Kids Hostel	Selangor	2021-12-15	1500000
KL200	100	150	KL Inner City Road Expansion	Kuala Lumpur	2022-01-20	4500000
KL100	110	130	Putra Jaya Raya	Kuala Lumpur	2022-03-10	12000000
PG300	130	110	Jawi River Bridge	Penang	2022-05-05	350000
JH200	NULL	150	Causeway Lighting	Johor	2022-05-25	250000

GROUP BY

- The GROUP BY clause is a SQL command that is used to **group rows that have the same values**.
- The GROUP BY clause is used in the SELECT statement.
- Optionally it is used in conjunction with aggregate functions to produce summary reports from the database.
- That's what it does, **summarizing data** from the database.
- The queries that contain the GROUP BY clause are called grouped queries and only return a single row for every grouped item.

GROUP BY examples

```
select location, count(*) as projectcount from project
group by location
```

%

Results | Messages

location	projectcount
Johor	2
Kuala Lumpur	2
Penang	1
Selangor	1

GROUP BY examples

```
SELECT [ProjectID], count([EmpID]) as EmployeeCount
FROM [AssignedEmployees]
Group By ProjectID
```

Results | Messages

	ProjectID	EmployeeCount
1	KL100	2
2	KL200	2
3	SEL200	1

GROUP BY examples

```
select clientid, count(*) as projectcount,
sum(value) as totalvalue
from project
group by clientid
```

Results | Messages

clientid	projectcount	totalvalue
100	1	1500000
110	1	350000
130	1	12000000
150	3	7750000

Aggregate Functions

- SUM – returns the sum of the column values
- MIN – returns the minimum value
- MAX – returns the maximum value
- AVG – returns the average value
- COUNT – returns the number of records

SUM, MIN, MAX, AVG, COUNT

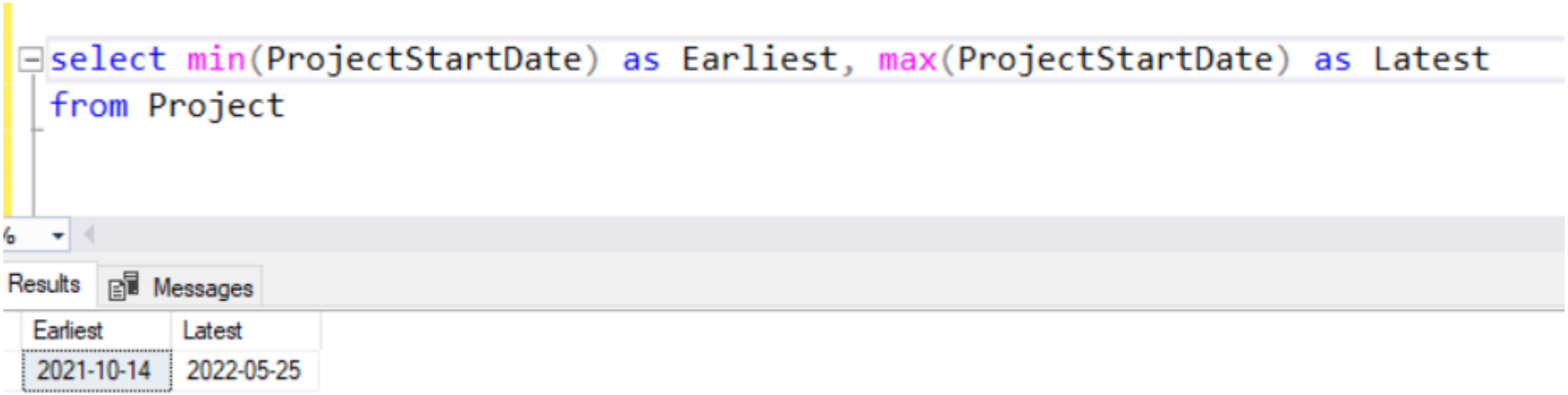
ProjectID	ProjectMgr	ClientID	ProjectName	Location	ProjectStartDate	Value
JH200	NULL	150	Causeway Lighting	Johor	2022-05-25	250000
JH400	NULL	150	Community Housing	Johor	2021-10-14	3000000
KL100	110	130	Putra Jaya Raya	Kuala Lumpur	2022-03-10	12000000
KL200	100	150	KL Inner City Road Expansion	Kuala Lumpur	2022-01-20	4500000
PG300	130	110	Jawi River Bridge	Penang	2022-05-05	350000
SEL200	120	100	Kids Hostel	Selangor	2021-12-15	1500000

```
select count(*) as [Total Number of Projects] ,
       sum(Value) as [Total Value],
       avg(Value) as [Average Value Per Project],
       min(Value) as [Cheapest Project],
       max(Value) as [Highest Value]
from Project
```

Results | Messages

Total Number of Projects	Total Value	Average Value Per Project	Cheapest Project	Highest Value
6	21600000	3600000.000000	250000	12000000

MIN, MAX



```
select min(ProjectStartDate) as Earliest, max(ProjectStartDate) as Latest
from Project
```

Results | Messages

Earliest	Latest
2021-10-14	2022-05-25

SubQuery

- A Subquery or Inner query or a Nested query is **a query within another SQL query and embedded within the WHERE clause**.
- A subquery is used to return data that will be used in the main query as a condition to further restrict the data to be retrieved.

SubQuery examples

- Get the projects which has at least one employee assigned to it

```
select projectname from project
where projectid in
(select projectid from assignedemployees)
```

esults | Messages

projectname
Putra Jaya Raya
KL Inner City Road Expansion
Kids Hostel

Subquery examples

- Generate a list of employees who are assigned to projects

```
select * from employee
where empid in
(select empid from assignedemployees)
```

Results | Messages

EmpID	EmpName	EmailID
100	Nathan	nathan@abc.com
120	Jason	jason@abc.com
130	Sharon	sharon@abc.com
140	Tyson	tyson@abc.com

Subquery examples

- Generate a list of employees who are NOT assigned to projects

```
select * from employee
where empid NOt in
(select empid from assignedemployees)
```

Results | Messages

EmpID	EmpName	EmailID
110	Ismail	ismail@abc.com

Subquery examples

- Identify projects that has value more than average project value

```
select projectname, value
from project
where value >
(select avg(value) from project)
```

Results | Messages

projectname	value
Putra Jaya Raya	12000000
KL Inner City Road Expansion	4500000

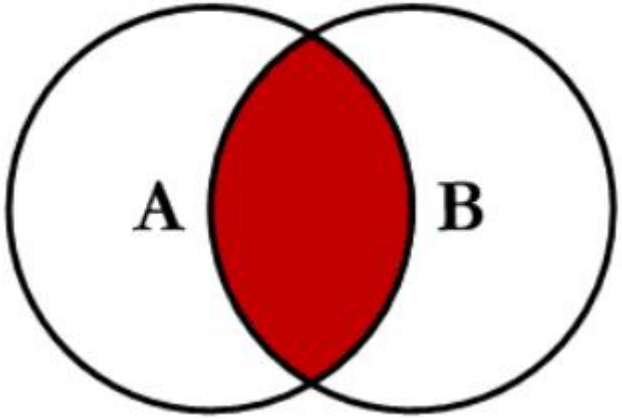
What is SQL JOIN ?

A · P · U
ASIA PACIFIC UNIVERSITY
OF TECHNOLOGY & INNOVATION

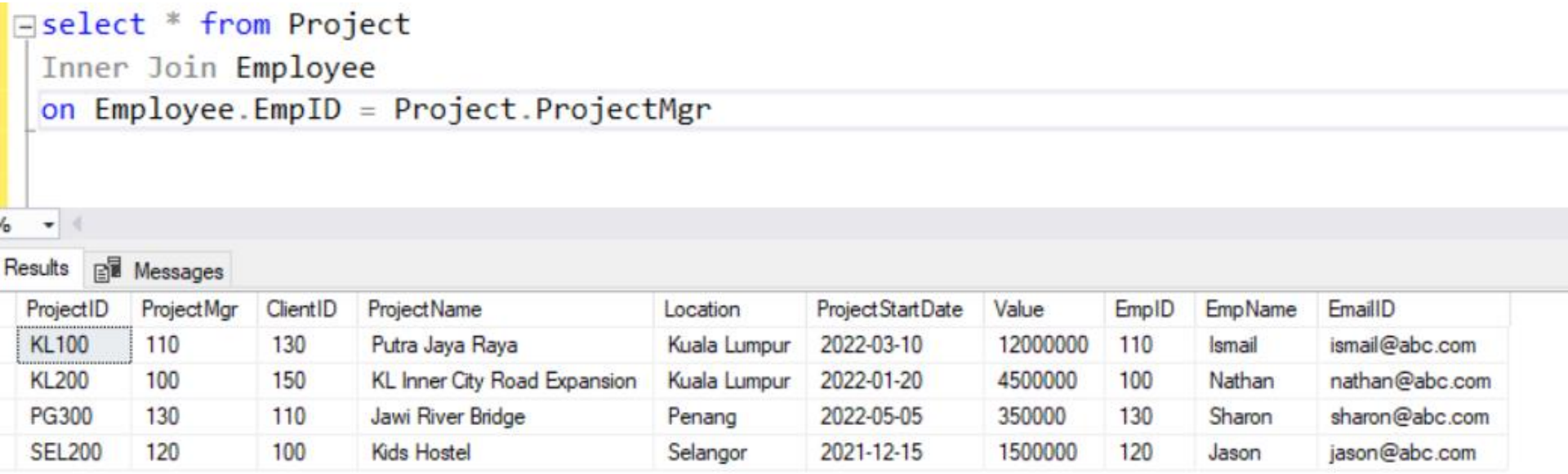
- JOINS in SQL are commands which are used to **combine rows from two or more tables**, based on a related column between those tables.

Inner Join

- Inner join produces the intersection between 2 sets
- Produces the set of records that match in both Table A and Table b



Sample Data from Inner Join



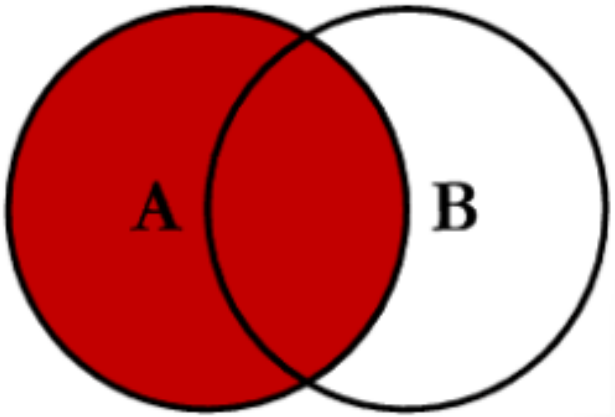
```
select * from Project
Inner Join Employee
on Employee.EmpID = Project.ProjectMgr
```

Results | Messages

ProjectID	ProjectMgr	ClientID	ProjectName	Location	ProjectStartDate	Value	EmpID	EmpName	EmailID
KL100	110	130	Putra Jaya Raya	Kuala Lumpur	2022-03-10	12000000	110	Ismail	ismail@abc.com
KL200	100	150	KL Inner City Road Expansion	Kuala Lumpur	2022-01-20	4500000	100	Nathan	nathan@abc.com
PG300	130	110	Jawi River Bridge	Penang	2022-05-05	350000	130	Sharon	sharon@abc.com
SEL200	120	100	Kids Hostel	Selangor	2021-12-15	1500000	120	Jason	jason@abc.com

Left Outer Join

- Left outer join produces a complete set of records from Table A, with the matching records (where available) in Table B.
- If there is no match, the right side will contain null.



Sample Data from Left Outer Join

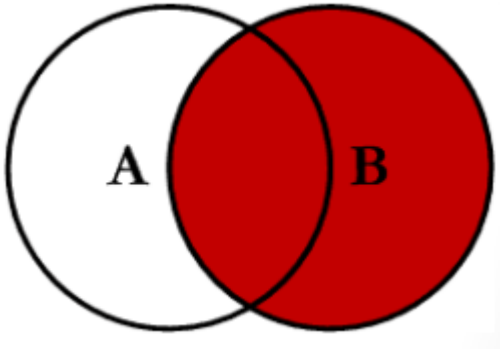

```
select * from Employee
left Outer Join AssignedEmployees
on Employee.EmpID = AssignedEmployees.EmpID
```

Results | Messages

EmpID	EmpName	EmailID	EmpID	ProjectID
100	Nathan	nathan@abc.com	100	KL100
100	Nathan	nathan@abc.com	100	KL200
110	Ismail	ismail@abc.com	NULL	NULL
120	Jason	jason@abc.com	120	KL100
130	Sharon	sharon@abc.com	130	KL200
140	Tyson	tyson@abc.com	140	SEL200

Right Outer Join

- Right outer join produces a complete set of records from Table B, with the matching records (where available) in Table A.



- If there is no match, the left side will contain null.

Sample Data from Right Outer Join

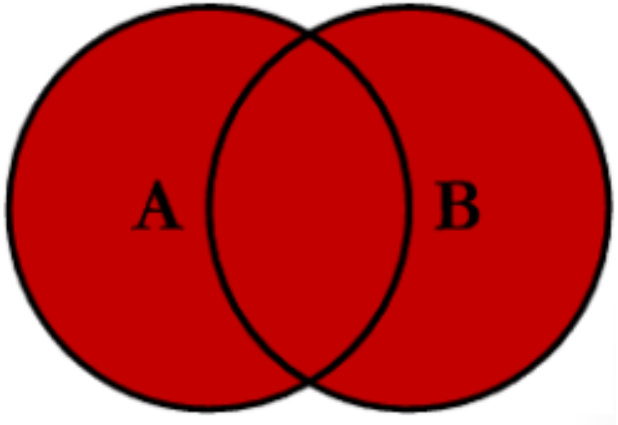

```
select * from AssignedEmployees
right Outer Join Project
on Project.ProjectID = AssignedEmployees.ProjectID
```

Results | Messages

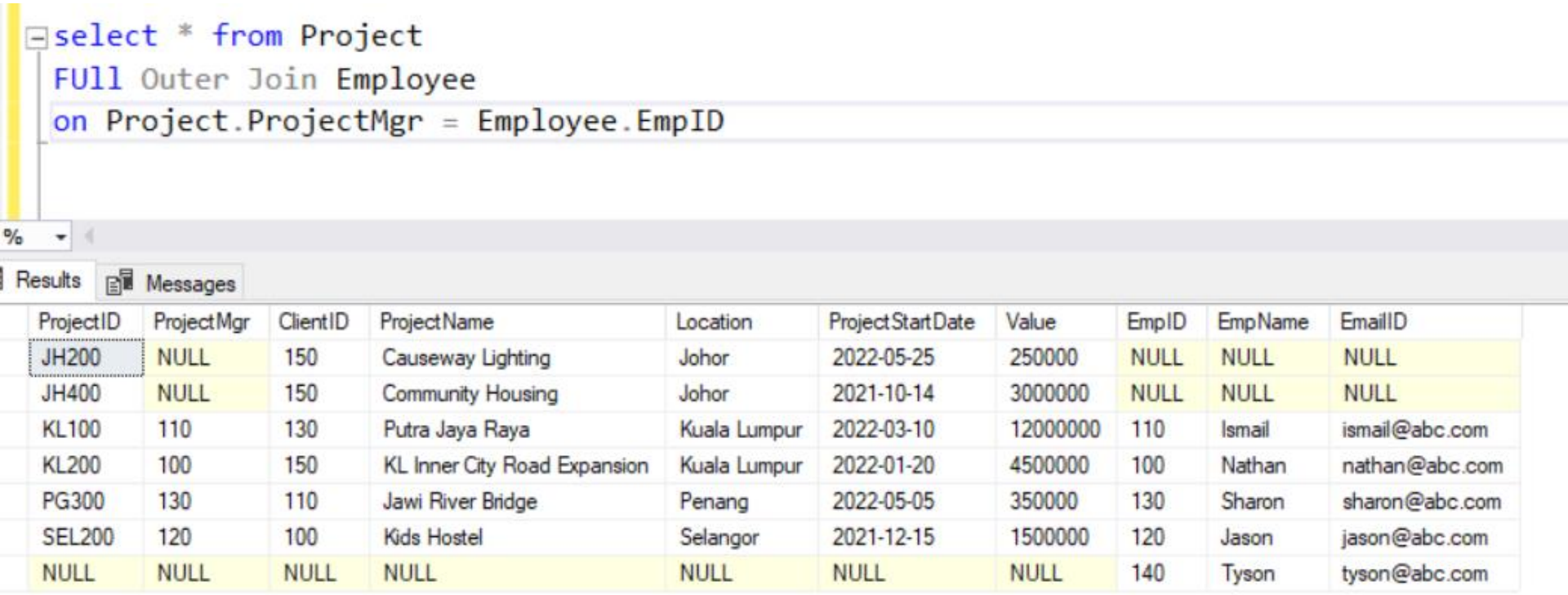
EmpID	ProjectID	ProjectID	ProjectMgr	ClientID	ProjectName	Location	ProjectStartDate	Value
NULL	NULL	JH200	NULL	150	Causeway Lighting	Johor	2022-05-25	250000
NULL	NULL	JH400	NULL	150	Community Housing	Johor	2021-10-14	3000000
100	KL100	KL100	110	130	Putra Jaya Raya	Kuala Lumpur	2022-03-10	12000000
120	KL100	KL100	110	130	Putra Jaya Raya	Kuala Lumpur	2022-03-10	12000000
130	KL200	KL200	100	150	KL Inner City Road Expansion	Kuala Lumpur	2022-01-20	4500000
100	KL200	KL200	100	150	KL Inner City Road Expansion	Kuala Lumpur	2022-01-20	4500000
NULL	NULL	PG300	130	110	Jawi River Bridge	Penang	2022-05-05	350000
140	SEL200	SEL200	120	100	Kids Hostel	Selangor	2021-12-15	1500000

Full Outer Join

- Full outer join produces the set of all records in Table A and Table B, with matching records from both sides where available.
- If there is no match, the missing side will contain null.



Sample Data from Full Outer Join



```
select * from Project
FUll Outer Join Employee
on Project.ProjectMgr = Employee.EmpID
```

Results | Messages

ProjectID	ProjectMgr	ClientID	ProjectName	Location	ProjectStartDate	Value	EmpID	EmpName	EmailID
JH200	NULL	150	Causeway Lighting	Johor	2022-05-25	250000	NULL	NULL	NULL
JH400	NULL	150	Community Housing	Johor	2021-10-14	3000000	NULL	NULL	NULL
KL100	110	130	Putra Jaya Raya	Kuala Lumpur	2022-03-10	12000000	110	Ismail	ismail@abc.com
KL200	100	150	KL Inner City Road Expansion	Kuala Lumpur	2022-01-20	4500000	100	Nathan	nathan@abc.com
PG300	130	110	Jawi River Bridge	Penang	2022-05-05	350000	130	Sharon	sharon@abc.com
SEL200	120	100	Kids Hostel	Selangor	2021-12-15	1500000	120	Jason	jason@abc.com
NULL	NULL	NULL	NULL	NULL	NULL	NULL	140	Tyson	tyson@abc.com

Generate a list projects assigned with project manager. Show project name, client name and project manager name only

```
select projectname as [Project name],
       clientname as [CLient Name],
       empname as [Project Manager]
from Project
Inner Join Client on Project.ClientID = Client.ClientID
Inner Join Employee on Employee.EmpID = Project.ProjectMgr
```

Results | Messages

Project name	CLient Name	Project Manager
Putra Jaya Raya	Joe	Ismail
KL Inner City Road Expansion	Jason	Nathan
Jawi River Bridge	Au	Sharon
Kids Hostel	Ramli	Jason

Generate a list of the projects. Show project name, client name and project manager name (if any) only

```
select projectname as [Project name],
       clientname as [CLient Name],
       empname as [Project Manager]
from Project
Inner Join Client on Project.ClientID = Client.ClientID
Left Outer Join Employee on Employee.EmpID = Project.ProjectMgr
```

Results | Messages

Project name	CLient Name	Project Manager
Causeway Lighting	Jason	NULL
Community Housing	Jason	NULL
Putra Jaya Raya	Joe	Ismail
KL Inner City Road Expansion	Jason	Nathan
Jawi River Bridge	Au	Sharon
Kids Hostel	Ramli	Jason

List down all projects. Show any employee id if they are assigned to it.

```
select projectname, empid from project
left outer join assignedemployees
on project.projectid = assignedemployees.projectid
```

Results | Messages

projectname	empid
Causeway Lighting	NULL
Community Housing	NULL
Putra Jaya Raya	100
Putra Jaya Raya	120
KL Inner City Road Expansion	130
KL Inner City Road Expansion	100
Jawi River Bridge	NULL
Kids Hostel	140

Question and Answer Session

A·P·U
ASIA PACIFIC UNIVERSITY
OF TECHNOLOGY & INNOVATION

Q & A

Next Topics: Security Requirements

- Until now, we have covered, the basics of database data requirements, design and development.
- Next, we will cover security requirements and implementation
- Security requirements are CIA
  - Confidentiality – User Management (who can access and who can do what), Encryption (hiding data from unauthorized people)
  - Integrity – Auditing (knowing what has happened) and Ensure Data Integrity using Built-In Functions (covered today) and Trigger (powerful SQL feature that has many uses)
  - Availability – Backup & Restore (ensure data is available to authorized users on a timely manner)