

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  | **XYZ Software Solutions**  23/12/2018 | |  | | |
|  |  |  | | |  | |
|  |  | Trinity Database  Institutional record management system | | |  | |
|  |  | | | | |  |
|  | **Introduction**  Trinity Music School based in Colombo currently practices disorganized and out of date, data processing methods in order to meet the demands in the growth of the school and flexibility, the school has request XYZ Software Solutions to create a relational database system to completely replace their school based record management system.  This report will attempt to display and explain certain SQL queries, provide evidence user interfaces, outputs and data validations.  Author: K.P.I. Shenesh Perera  IDM | | | | |  |
|  |  | |  | | | |

**Contents**

[Structured Query Language (SQL) 3](#_Toc1504529)

[Microsoft: SQL Server (v11.0.2100) 4](#_Toc1504530)

[Process of the creation of Trinity’s Database 5](#_Toc1504531)

[Creating & Accessing Trinity DB 5](#_Toc1504532)

[Creating the class table & inserting data 6](#_Toc1504533)

[Class Interface 6](#_Toc1504534)

[Creating the supplier table & inserting data 6](#_Toc1504535)

[Supplier Interface 7](#_Toc1504536)

[Creating the instrument table & inserting data 8](#_Toc1504537)

[Instrument Interface 8](#_Toc1504538)

[Creating the student table & inserting data 9](#_Toc1504539)

[Student Interface 9](#_Toc1504540)

[Creating the teacher table & inserting data 10](#_Toc1504541)

[Creating the attendance table & inserting data 10](#_Toc1504542)

[Creating the attendance table & inserting data 10](#_Toc1504543)

[Query Demonstration 11](#_Toc1504544)

[Task 1: Get details of a student registered in a particular month along with the id of the student (number of registration) 11](#_Toc1504545)

[Task 2: Total revenue from a student in a given month 12](#_Toc1504546)

[Task 3: Student details alongside the instrument they play 13](#_Toc1504547)

[Task 4: Filter student details based on the registration date 14](#_Toc1504548)

[Task 5: Show only part time classes 14](#_Toc1504549)

# Structured Query Language (SQL)

Pronounced Sequel or “es-kue-el” is the language by which machines converse with databases. According to the American National Standards Institute, SQL is the standard for interactions with Relational Database Management Systems. One of the most basic yet powerful features of SQL is to allowing developers to perform Create, Read, Update and Delete (CRUD) actions on a database.

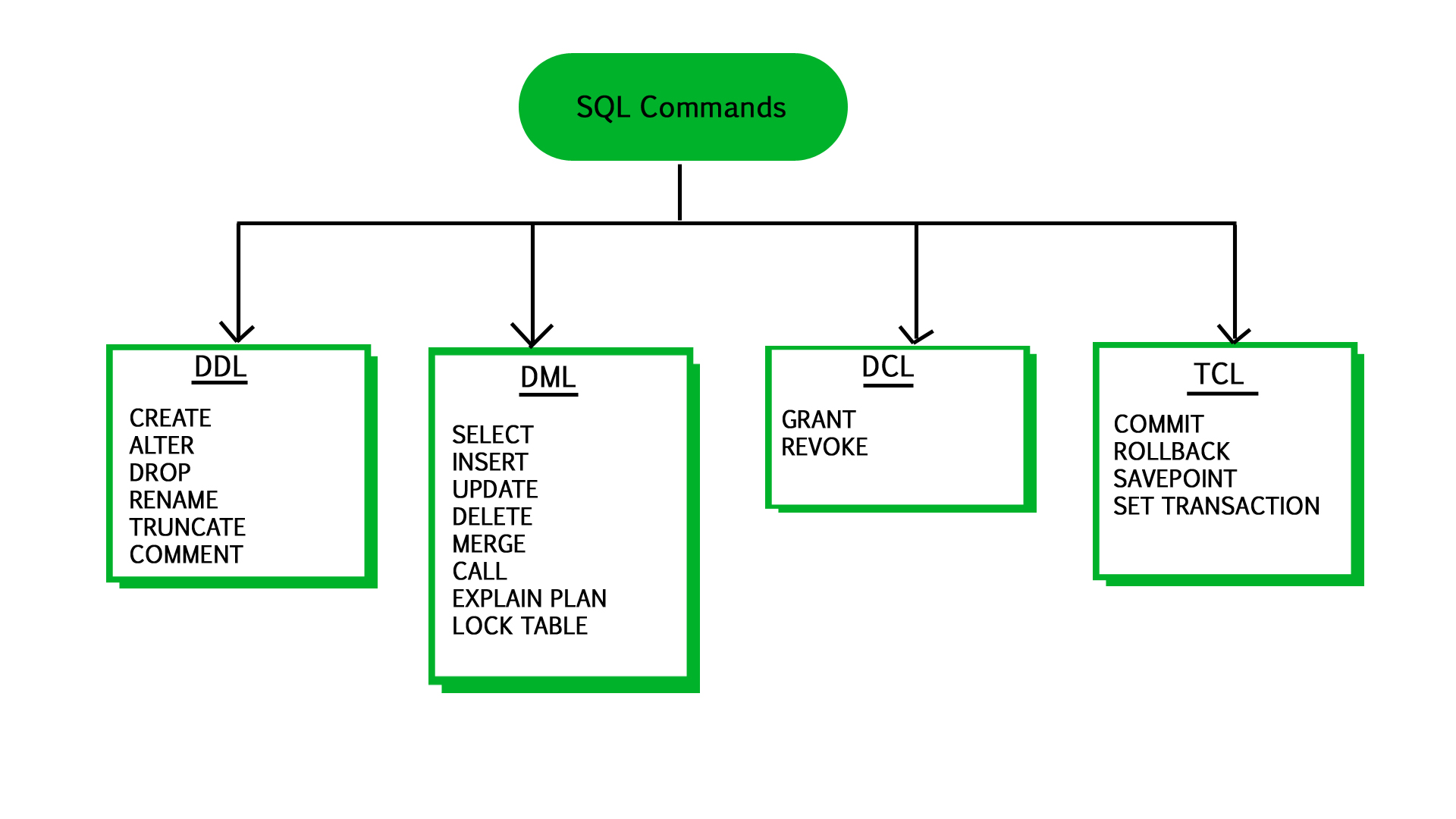
When it comes to Database Management Systems, there are 4 key languages that allow valid expression of queries and structure data stored within a database.

They are:

1. Data Definition Language: Defines the structure of data and ensures that data is stored in ways patterns can be identified. DDL statements enable definition of the metadata of a database. Creating schemas, tables, indexes and constraints within a database are all done using DDL.
2. Data Manipulation Language: Allows accessing and manipulating data that is within a database. Whenever a machine requires accessing something stored in a database, it must do so through an implemented layer of DML.
3. Data Control Language: This language has a lot of similarities with DML, but the key difference is that DCL allows the control of permissions to a database, ensuring access hierarchies and also provide the functions of DML. Execution of DCL statements is heavily transactional, and also consists of rollback parameters.
4. Transaction Control Language: Persisting changes made to the database is done solely by this language. Each particular statement that is executed to affect the database can be visualized as a “commit” and saving or undoing these commits is what TCL does.

The Structured Query Language incorporates the implementation of all these 4 languages used in Database Management Systems. As such it is highly important in the process of using databases for storage purposes. The following image shows how each of these 4 languages have been envisioned in SQL using commands.

Figure 1.0, [www.geeksforgeeks.org](http://www.geeksforgeeks.org), shubham\_tyagi4, 2016/05/24



# Microsoft: SQL Server (v11.0.2100)

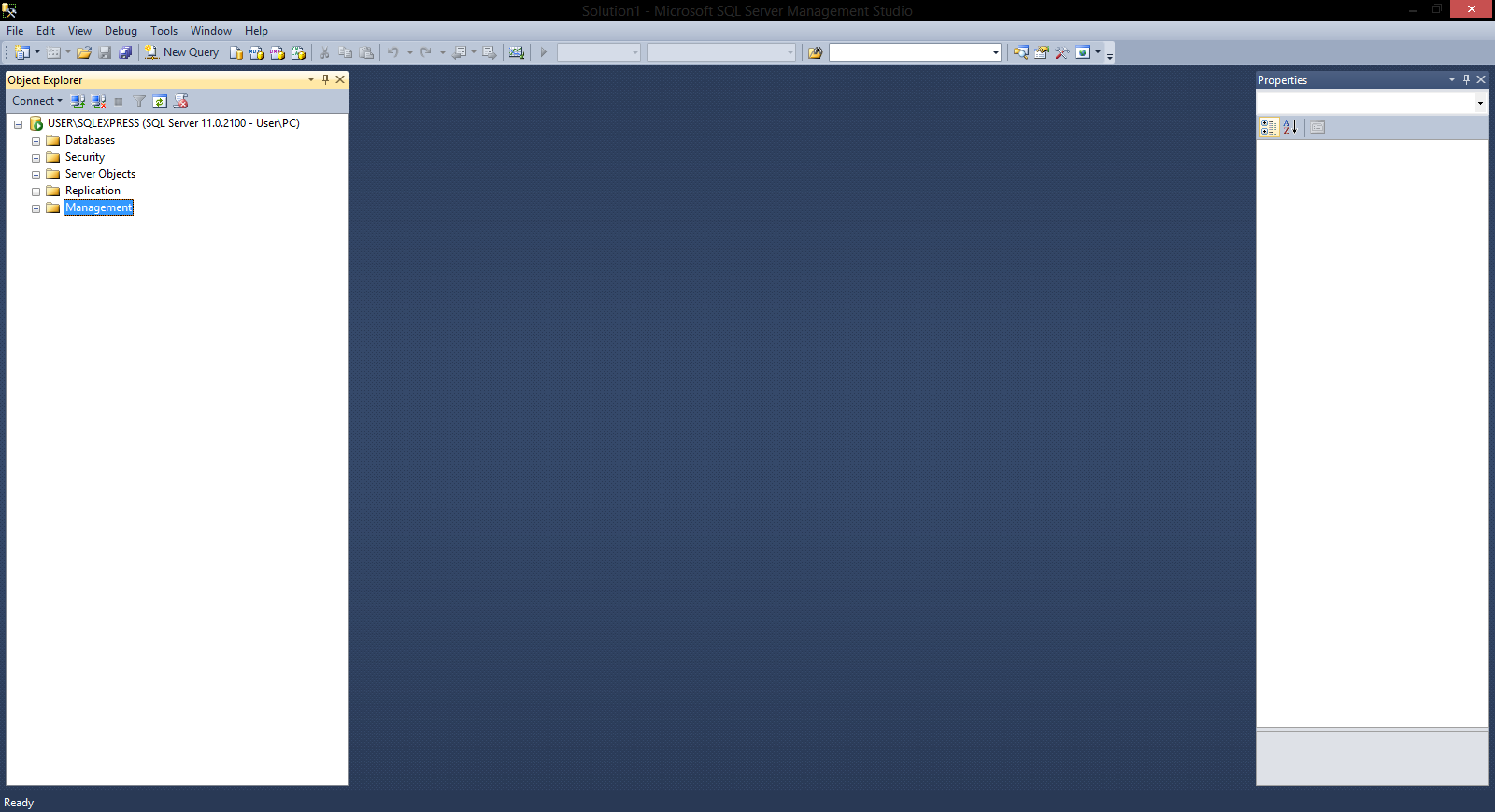
Microsoft SQL Server is a Relational Database Management System that was used in order to implement the Trinity’s relational database system. SQL Server supports a wide range of transaction processing, business intelligence & analytical interfaces to the IT industry. It is among the top 3 most used and most marketed database technologies out there.

SQL Server is built on top of a layer of SQL, as previously mentioned a standardized language used to query databases. The SQL implementation that SQL Server is built upon is known as Transact-SQL (T-SQL) and implementation produced by Microsoft, that adds an additional set of extensions to the standard Structured Query Language.

The SQL Server Management Studio was heavily used in the creation of the Trinity Database.

Following is an image of how SQL Server’s Management Studio’s main interface looks like upon connecting to the server.

Figure 1.1, Shenesh Perera, 22/12/2018



It has a very user friendly and organized look and is extremely robust in handling even the biggest challenges faced in the development of relational databases.

# Process of the creation of Trinity’s Database

The following attempt will be a series of steps that will go from Database creation to creating tables and then finally inserting data, alongside justification and data validation will be done.

## Creating & Accessing Trinity DB

CREATE DATABASE trinityHS

USE trinityHS

## 

By running these 2 queries, one after the other a database called “trinityHS” is created and then the Database Management System will be prompted to use “trinityHS” as the current active database. All of the following queries will therefore be performed on database “trinityHS”

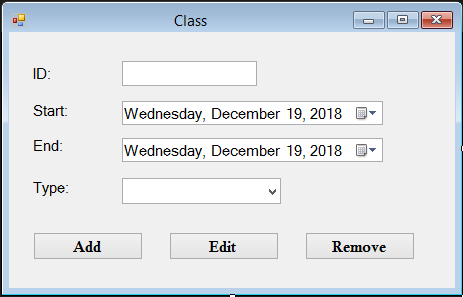
## Creating the class table & inserting data

|  |  |
| --- | --- |
| **Create Table** | **Insert Data** |
| CREATE TABLE class (  c\_id int PRIMARY KEY,  c\_start datetime,  c\_end datetime,  c\_type varchar (50)) | INSERT INTO class VALUES (1111, '20181018 08:00:00 AM', '20181018 01:34:09 PM', 'FULL\_TIME')  INSERT INTO class VALUES (1112, '20181019 08:15:00 AM', '20181018 02:00:00 PM', 'PART\_TIME')  INSERT INTO class VALUES (1113, '20181020 09:30:00 AM', '20181018 02:11:00 PM', 'PART\_TIME\_EVENING') |

Now a table called class is created, with 4 columns c\_id, c\_start, c\_end and c\_type with datatypes of Integer, datetime, datetime and varchar respectively. The primary key is c\_id and hence is naturally UNIQUE and NOT NULL. In the Insert query only the data of the data types that obeys the table’s schema is entered so the data is valid as such both the queries pass.

### Class Interface

Figure 1.3, Shenesh Perera 19/12/2018



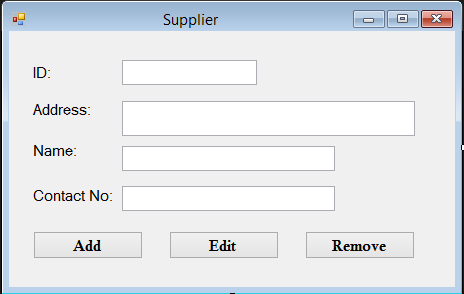
## Creating the supplier table & inserting data

|  |  |
| --- | --- |
| **Create Table** | **Insert Data** |
| CREATE TABLE supplier (  supp\_id int IDENTITY(100, 20) PRIMARY KEY,  supp\_address varchar(80),  supp\_name varchar(100),  supp\_contact\_no varchar(20)  ) | INSERT INTO supplier VALUES ('St Joseph''s St, Negombo', 'Yamaha Dealer', '031892834') |

Then a table called supplier is created, with 4 columns supp\_id, supp\_address, supp\_name and supp\_contact\_no with datatypes of Integer and the rest varchar respectively. The primary key is supp\_id and is automatically generated for convenience sake, the initial value is 100 and it increments by 20 with each entry. In the Insert query only the data of the data types that obeys the table’s schema is entered so the data is valid as such both the queries pass.

### Supplier Interface

Figure 1.4, Shenesh Perera,19/12/2018



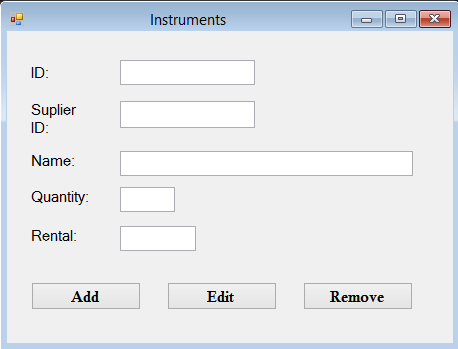
## Creating the instrument table & inserting data

|  |  |
| --- | --- |
| **Create Table** | **Insert Data** |
| CREATE TABLE instrument (  i\_id int PRIMARY KEY,  supp\_id int FOREIGN KEY REFERENCES supplier(supp\_id),  i\_name varchar(100),  i\_quantity int,  i\_rental money) | INSERT INTO instrument VALUES (103, 100, 'Violin', 13, 20398.00)  INSERT INTO instrument VALUES (102, 100, 'Guitar', 10, 15201.00) |

Now a table called instrument is created, with 5 columns i\_id, supp\_id, i\_name, i\_quantity, i\_rental with datatypes of Integer, Integer, varchar, Integer and money respectively. The primary key is i\_id and hence is naturally UNIQUE and NOT NULL. There is a foreign key in this table that references to the supplier table’s supp\_id column, so an insert query would only pass if the data entered to the supp\_id column also exists in the supp\_id column in the supplier table. In the Insert query only the data of the data types that obeys the table’s schema is entered so the data is valid and the data entered for the supp\_id column exists in the supp\_id column of the supplier table as well so both the queries pass.

### Instrument Interface

Figure 1.5, Shenesh Perera, 19/12/2018

****

## Creating the student table & inserting data

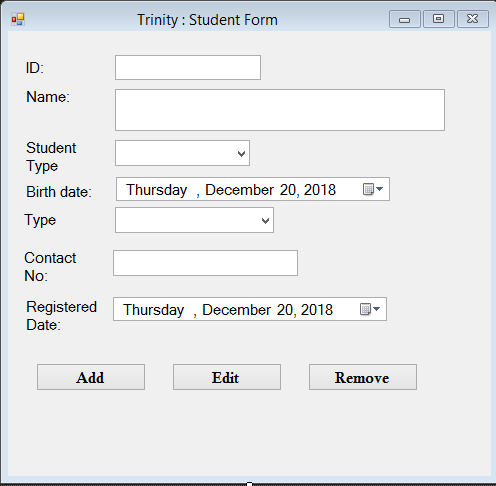
|  |  |
| --- | --- |
| **Create Table** | **Insert Data** |
| CREATE TABLE student (  s\_id int PRIMARY KEY,  i\_id int FOREIGN KEY REFERENCES instrument(i\_id),  s\_name varchar(100),  s\_type varchar(50),  s\_regist\_date datetime,  s\_dob datetime,  s\_contact\_no varchar(20) ) | INSERT INTO student VALUES (1, 103, 'Saline Man', 'FULL\_TIME', '20181018 00:00:00 AM', '20000928 00:00:00 AM', '076 1234567')  INSERT INTO student VALUES (2, 103, 'Saline Woman', 'PART\_TIME', '20181012 00:00:00 AM', '20000101 00:00:00 AM', '077 1234567')  INSERT INTO student VALUES (3, 103, 'Orange Man', 'PART\_TIME\_EVENING', '20181013 00:00:00 AM', '20000401 00:00:00 AM', '078 1234567')  INSERT INTO student VALUES (4, 102, 'Orange Woman', 'PART\_TIME\_EVENING', '20180715 00:00:00 AM', '20000602 00:00:00 AM', '079 1234567')  INSERT INTO student VALUES (5, 102, 'Banana Man', 'FULL\_TIME', '20181018 00:00:00 AM', '20000516 00:00:00 AM', '032 1234567') |

Now a table called student is created, with 7 columns s\_id, i\_id, s\_name, s\_type, s\_regist\_date, s\_dob, s\_contact\_no with datatypes of Integer, Integer, varchar, varchar, datetime, datetime and varchar respectively. The primary key is s\_id and hence is naturally UNIQUE and NOT NULL. There is a foreign key in this table that references to the instrument table’s i\_id column, so an insert query would only pass if the data entered to the i\_id column also exists in the i\_id column in the instrument table. In the Insert query only the data of the data types that obeys the table’s schema is entered so the data is valid and the data entered for the i\_id column exists in the i\_id column of the instrument table as well so both the queries pass.

**NOTE: From here on out, interface will no longer be visualized as I have included enough pictures to get the general idea of how each interface looks like.**

### Student Interface

Figure 1.6, Shenesh Perera,19/12/2018



## Creating the teacher table & inserting data

|  |  |
| --- | --- |
| **Create Table** | **Insert Data** |
| CREATE TABLE teacher (  t\_id int PRIMARY KEY,  t\_name varchar(100),  t\_contact\_no varchar(20) ) | INSERT INTO teacher VALUES (10001, 'Dumbledore McField', '079 123467') |

Now a table called teacher is created, with 3 columns t\_id, t\_name and t\_contact\_no with datatypes of Integer, varchar and varchar respectively. The primary key is t\_id and hence is naturally UNIQUE and NOT NULL. In the Insert query only the data of the data types that obeys the table’s schema is entered so the data is valid so both the queries pass.

## Creating the attendance table & inserting data

|  |  |
| --- | --- |
| **Create Table** | **Insert Data** |
| CREATE TABLE attendance (  a\_index int IDENTITY(10, 5) PRIMARY KEY,  c\_id int FOREIGN KEY REFERENCES class(c\_id),  s\_id int FOREIGN KEY REFERENCES student(s\_id),  attend\_time datetime NOT NULL,  daily\_cost money,  ) | INSERT INTO attendance VALUES (1111, 1, '20181018 08:05:43 AM', 2000.00)  INSERT INTO attendance VALUES (1111, 2, '20181018 07:39:50 AM', 3000.00) |

Now a table called attendance is created, with 4 columns c\_id, s\_id, attend\_time and daily\_cost with datatypes of Integer, Integer, datetime and money. In this table there are 2 foreign keys, c\_id and s\_id from class and student tables respectively as the 2 tables are engaged in a many-to-many relationship. The primary key is a\_index and all attributes depend on the index and so are UNIQUE and NOT NULL. In the Insert query only the data of the data types that obeys the table’s schema is entered so the data is valid so both the queries pass.

## Creating the attendance table & inserting data

|  |  |
| --- | --- |
| **Create Table** | **Insert Data** |
| CREATE TABLE class\_subjects\_linked (  cs\_index int IDENTITY(10, 5) PRIMARY KEY,  c\_id int FOREIGN KEY REFERENCES class(c\_id),  t\_id int FOREIGN KEY REFERENCES teacher(t\_id),  c\_subject varchar(50) ) | INSERT INTO class\_subjects\_linked VALUES (1111, 10001, 'Cleaning Instruments')  INSERT INTO class\_subjects\_linked VALUES (1111, 10001, 'Cleaning Instruments') |

Now a table called class\_subjects\_linked is created, with 3 columns c\_id, t\_id and c\_subject with datatypes of Integer, Integer and varchar. In this table there are 2 foreign keys, c\_id and 5\_id from class and teacher tables respectively as the 2 tables are engaged in a many-to-many relationship. The primary key is cs\_index and all attributes depend on it, and is UNIQUE and NOT NULL. In the Insert query only the data of the data types that obeys the table’s schema is entered so the data is valid so both the queries pass.

# Query Demonstration

The problems declared within the document will now be evaluated, then queries will be written and the output of the action of those queries on the database will be shown.

## Task 1: Get details of a student registered in a particular month along with the id of the student (number of registration)

First a query must be run in order to see if there is data available in the student table that can be used to run the query declared by the task, therefore the following query is run:

SELECT \* FROM student

The following is the output that was obtained:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | 103 | Saline Man | FULL\_TIME | 2018-10-18 00:00:00.000 | 2000-09-28 00:00:00.000 | 076 1234567 |
| 2 | 103 | Saline Woman | PART\_TIME | 2018-10-12 00:00:00.000 | 2000-01-01 00:00:00.000 | 077 1234567 |
| 3 | 103 | Orange Man | PART\_TIME\_EVENING | 2018-10-13 00:00:00.000 | 2000-04-01 00:00:00.000 | 078 1234567 |
| 4 | 102 | Orange Woman | PART\_TIME\_EVENING | 2018-07-15 00:00:00.000 | 2000-06-02 00:00:00.000 | 079 1234567 |
| 5 | 102 | Banana Man | FULL\_TIME | 2018-10-18 00:00:00.000 | 2000-05-16 00:00:00.000 | 032 1234567 |

Now, to satisfy the task I will get the ID, registration date and type of the students that have been registered on the 10th month of 2018, so the following query is run:

SELECT

s\_id, s\_regist\_date, s\_type

FROM student

WHERE MONTH(s\_regist\_date) = 10

The following output is obtained:

|  |  |  |
| --- | --- | --- |
| 1 | 2018-10-18 00:00:00.000 | FULL\_TIME |
| 2 | 2018-10-12 00:00:00.000 | PART\_TIME |
| 3 | 2018-10-13 00:00:00.000 | PART\_TIME\_EVENING |
| 5 | 2018-10-18 00:00:00.000 | FULL\_TIME |

Upon analysis one can see, that the output satisfies the task as the 4th student is registered on the 7th month of 2018 while the rest have been registered on the 10th.

## Task 2: Total revenue from a student in a given month

First a query must be run in order to see if there is data available in the attendance table that can be used to run the query declared by the task, therefore the following query is run:

SELECT \* FROM attendance

The following output was obtained:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 10 | 1111 | 1 | 2018-10-18 08:05:43.000 | 2000.00 |
| 15 | 1111 | 1 | 2018-10-19 08:05:43.000 | 2000.00 |
| 20 | 1111 | 1 | 2018-10-20 08:05:43.000 | 2000.00 |
| 25 | 1111 | 2 | 2018-10-18 07:39:50.000 | 3000.00 |

Now to satisfy the task I will get the ID of the student and total income for the 10th month of 2018, so the following query is run:

SELECT

s\_id, SUM(daily\_cost) As total\_incoome

FROM attendance

WHERE MONTH(attend\_time) = 10

GROUP BY s\_id

The following output is obtained:

|  |  |
| --- | --- |
| 1 | 6000.00 |
| 2 | 3000.00 |

Upon analysis one can see that on the 10th month the student with ID 1 has made 3 payments each of 2000, which adds up to 6000 and the student with ID 2 has made only a payment of 3000.

## Task 3: Student details alongside the instrument they play

First 2 queries must be run in order to see if there is data available in the student table and in the instrument table that can be used to run the query declared by the task, therefore the following queries are run and their respective output is shown:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| SELECT \* FROM student | 1 | 103 | Saline Man | FULL\_TIME | 2018-10-18 00:00:00.000 | 2000-09-28 00:00:00.000 | 076 1234567 |
| 2 | 103 | Saline Woman | PART\_TIME | 2018-10-12 00:00:00.000 | 2000-01-01 00:00:00.000 | 077 1234567 |
| 3 | 103 | Orange Man | PART\_TIME\_EVENING | 2018-10-13 00:00:00.000 | 2000-04-01 00:00:00.000 | 078 1234567 |
| 4 | 102 | Orange Woman | PART\_TIME\_EVENING | 2018-07-15 00:00:00.000 | 2000-06-02 00:00:00.000 | 079 1234567 |
| 5 | 102 | Banana Man | FULL\_TIME | 2018-10-18 00:00:00.000 | 2000-05-16 00:00:00.000 | 032 1234567 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| SELECT \* FROM instrument | 102 | 100 | Guitar | 10 | 15201.00 |
| 103 | 100 | Violin | 13 | 20398.00 |

Now in order to satisfy the task declared, I will run a query that will obtain s\_id, name of the instrument played by that student, s\_name and s\_type:

SELECT

student.s\_id, instrument.i\_name As instrument\_name, student.s\_name, student.s\_type

FROM student

INNER JOIN instrument

ON student.i\_id = instrument.i\_id

The following output is obtained:

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | Violin | Saline Man | FULL\_TIME |
| 2 | Violin | Saline Woman | PART\_TIME |
| 3 | Violin | Orange Man | PART\_TIME\_EVENING |
| 4 | Guitar | Orange Woman | PART\_TIME\_EVENING |
| 5 | Guitar | Banana Man | FULL\_TIME |

## Task 4: Filter student details based on the registration date

As the view of the data in student table has been previously visualized, I will directly query the database with the appropriate query to filter student details on the registration date of 18th of October 2018.

SELECT s\_name, s\_regist\_date, s\_type FROM student

WHERE s\_regist\_date = '20181018'

The following output is obtained:

|  |  |  |
| --- | --- | --- |
| Saline Man | 2018-10-18 00:00:00.000 | FULL\_TIME |
| Banana Man | 2018-10-18 00:00:00.000 | FULL\_TIME |

## Task 5: Show only part time classes

First a query must be run in order to see if there is data available in the class table that can be used to run the query declared by the task, therefore the following query is run:

SELECT \* FROM class

The following output is obtained:

|  |  |  |  |
| --- | --- | --- | --- |
| 1111 | 2018-10-18 08:00:00.000 | 2018-10-18 13:34:09.000 | FULL\_TIME |
| 1112 | 2018-10-19 08:15:00.000 | 2018-10-18 14:00:00.000 | PART\_TIME |
| 1113 | 2018-10-20 09:30:00.000 | 2018-10-18 14:11:00.000 | PART\_TIME\_EVENING |

In order to satisfy the task, I will get only c\_id and c\_type of the class table using the following query:

SELECT c\_id, c\_type

FROM class

WHERE c\_type = 'PART\_TIME'

The following output is obtained:

|  |  |
| --- | --- |
| 1112 | PART\_TIME |

As only class 1112 is of the PART\_TIME type.