**SQL Scripts**

**SLIDE 46**

Obtain all information on the Students not attending course 1

**select \* from uni.student**

**where oursed <> 1 ;Text, table

Description automatically generated**

Obtain the first name, surname and Date of Birth for the student with the email address: [val.bolger@example](mailto:val.bolger@example).com

**select forenames, surname, dateofbirth**

**from uni.student**

**where emailaddress = ‘val.bolger@example.com’;**

Table

Description automatically generated

Obtain a list of the modules which have the subject Economics

**select \* from uni.module**

**where subject = ‘Economics’ ;**

**Graphical user interface, application, table, Excel

Description automatically generated**

Obtain a list of class numbers and their dates which are scheduled before 21st September 2020

**select \* from uni.schedule**

**where cdate < ‘2020-09-21’**

Table

Description automatically generated

**SLIDE 48**

In the example database, write inserts to insert the following information. Screenshot the SQL query and the results obtained.

Insert a record for a new course named Deep-Space Radar Telemetry

write an insert to insert records for the following modules:

String Theory

Exotic Matter

Harnessing the Einstein-Rosen Bridge

Supercollision and miniature Black Holes

(these modules are worth 20 credits each, at level 6 and are taught on the Quantum Physics Course)

**INSERT INTO**

**uni.module(ModuleName, Subject , Level, Credits)**

**VALUES**

**('String Theory','Quantum Physics Course', '6', '20'),**

**('Exotic Matter','Quantum Physics Course', '6', '20'),**

**('Harnessing the Einstein-Rosen Bridge','Quantum Physics Course', '6', '20'),**

**('Supercollision','Quantum Physics Course', '6', '20'),**

**('miniature Black Holes','Quantum Physics Course', '6', '20');**

**select \* from uni.modules;**

**INSERT INTO uni.class(LecturerID , ModuleID)**

**VALUES**

**('6', '124 '),**

**('6', '125'),**

**('6', '126 '),**

**('6', '127 '),**

**('6', '128 ');**

**select \* from uni.class ;**

**Table

Description automatically generated**

**SLIDE 50**

Using the information from the previous example and the LecturerID of 6, create a class for each new module

Count how students are enrolled overall

**select count(\*) from uni.student**

Table

Description automatically generated

Calculate the sum of full time fees for every full-time course

**select sum(fulltimefee) from uni.fees**

Table

Description automatically generated

Identify the cost of the least and most expensive course

**select min(fulltimefee) from uni.fees;**

**Table

Description automatically generated**

**select max(fulltimefee) from uni.fees**

**Table

Description automatically generated**

Calculate the average cost of all part time courses

**select avg(parttimefee) from uni.fees ;**

Table

Description automatically generated

Calculate the fee of each full time course after applying (subtracting) the scholarship discount

**select**

**courseid,**

**fulltimefee-scholarshipdiscount**

**from uni.fees ;**

Table

Description automatically generated

Extension:

Select only the course number of the cheapest full-time course

**select**

**courseid**

**from**

**uni.fees**

**where**

**fulltimefee = (select min(fulltimefee) from uni.fees)**

Table

Description automatically generated

Find cost of the most expensive course after applying the scholarship discount

**select**

**courseid, fulltimefee-scholarshipdiscount**

**from**

**uni.fees**

**order by 2 desc**

Table

Description automatically generated

Count the number of applications for History courses made between 01/03/2020 and 30/08/2020

**select \* from uni.application**

**where**

**courseappliedfor = 11**

**and**

**dateofapplication**

**between '2020-03-01' and '2020-08-30'**

**Graphical user interface, text

Description automatically generated**

**SLIDE 53**

Obtain all the course information for courses with the CourseIDs of 1,3,5 and 7

**select \* from uni.course**

**where courseid in (1,3,5,7)**

Obtain a list of all modules taught on courses which have a Full Time Fee greater than 9000

**select \* from modules**

**where courseid in**

**(select courseid**

**from uni.fees**

**where fulltimefee > 9000)**

Obtain a list of classes for modules taught on courses which have a Full Time Fee greater than 9000

**select \* from uni.class where moduleid in**

**(select moduleid from uni.module**

**where courseid in**

**(select courseid**

**from uni.fees**

**where fulltimefee > 9000))**

Extension

Find a list of studentIDs for the latest class on the most expensive course

HINT: You may need an AND in your WHERE clause to solve this

**SLIDE 54**

Obtain a list of Students and the name of the Courses they are studying

**select student.\*, course.coursename**

**from**

**uni.student inner join uni.course**

**on**

**student.courseid = course.courseid**

Graphical user interface, text, application

Description automatically generated

Obtain a list of course names, full time fees and part time fees for each course

**select**

**course.coursename,**

**fees.fulltimefee,**

**fees.parttimefee**

**from uni.course inner join uni.fees**

**on course.courseid = fees.courseid**

Table

Description automatically generated