COMP508 DATABASE SYSTEM DESIGN

SEMESTER 1, 2024

Paired Assignment

Part A Entity Relationship Modelling and Logical Database Design

Task 1: Identify Entities & Develop Business Rules

Whangarei Institute of Technology Case Study

Entity Relationship Modelling and Logical Database Design

1. Programmes to ProgrammeCourses

- Each Programme must include one or more ProgrammeCourses.
- Each ProgrammeCourse must be part of one Programme.

2. ProgrammeCourses to Courses

- Each ProgrammeCourse must contain one Course.
- Each Course may be included in one or more ProgrammeCourses.

3. Courses to CourseCoordinators

- Each course is coordinated by one or more AcademicStaff.
- Each AcademicStaff coordinates one or more courses.

4. AcademicStaff to CourseTeaching

- Each AcademicStaff teaches one or more courses.
- Each Course may be taught by one or more AcademicStaff.

5. AcademicStaff to CourseCoordinators

- Each AcademicStaff Member may coordinate one or more Courses.
- Each Course may be coordinated by one or more AcademicStaff.

6. AcademicStaff to Departments

- Each AcademicStaff must belong to one Department.
- Each Department must have one or more AcademicStaff.

7. Students to NextOfKin

- Each Student may have one or more NextOfKin.
- Each NextOfKin must be related to one Student.

8. Students to StudentPerformance

- Each Student must have one or more StudentPerformance records.
- Each StudentPerformance record must belong to one Student.

9. StudentPerformance to Courses

- Each StudentPerformance record must be for one Course.
- Each Course may have one or more StudentPerformance records.

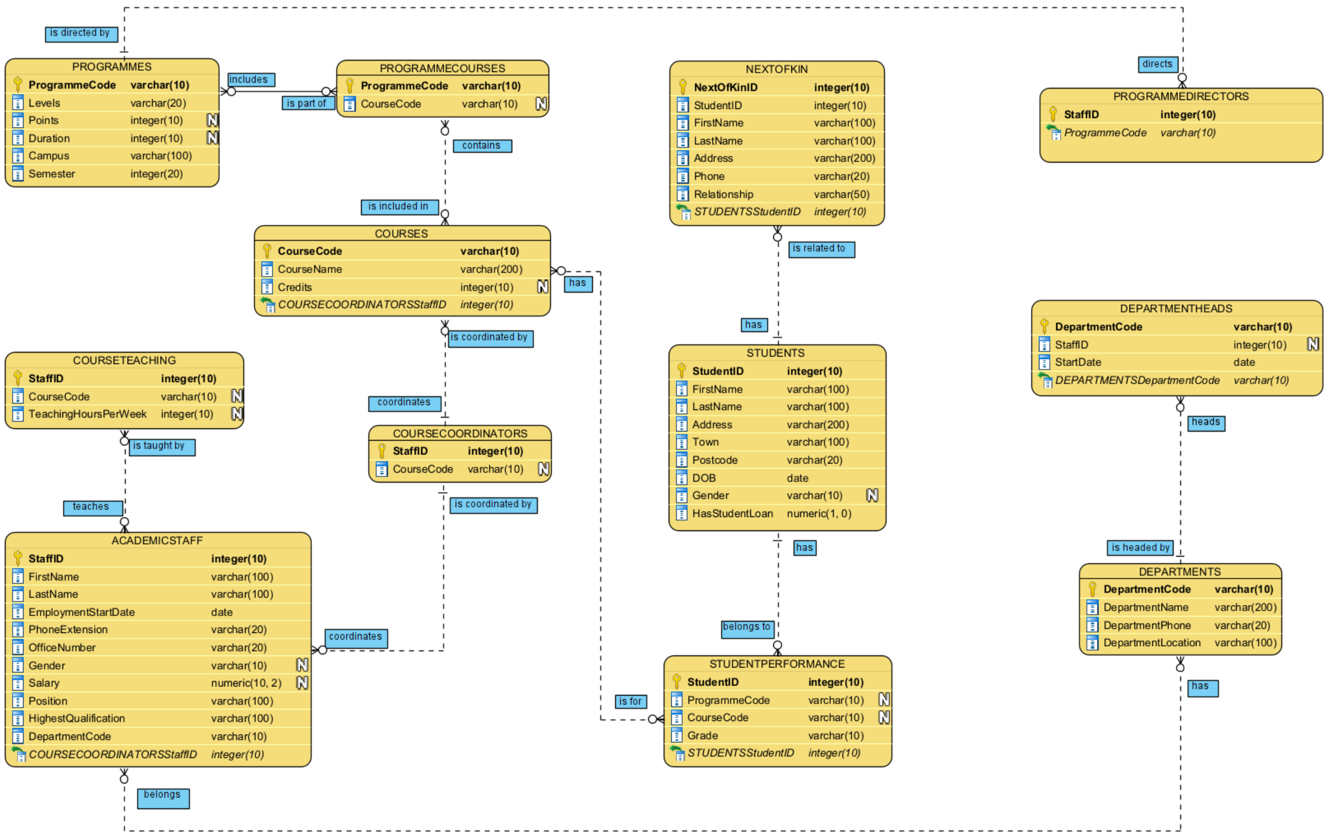
10. ProgrammeDirectors to Programmes

- Each Programme must be directed by one AcademicStaff.
- Each AcademicStaff may direct one or more Programmes.

11. DepartmentHeads to Departments

- Each Department must be headed by one AcademicStaff.
- Each AcademicStaff may head one Department.

Task 2: Logical Entity Relationship Diagram (ERD)



PROGRAMMES		
ProgrammeCode	**varchar(10)**	
Levels	varchar(20)	
Points	integer(10)	N
Duration	integer(10)	N
Campus	varchar(100)	
Semester	integer(20)	

PROGRAMMECOURSES		
ProgrammeCode	**varchar(10)**	
CourseCode	varchar(10)	N

COURSES		
CourseCode	**varchar(10)**	
CourseName	varchar(200)	
Credits	integer(10)	N
COURSECOORDINATORSStaffID	*integer(10)*	

COURSETEACHING		
StaffID	**integer(10)**	
CourseCode	varchar(10)	N
TeachingHoursPerWeek	integer(10)	N

COURSECOORDINATORS		
StaffID	**integer(10)**	
CourseCode	varchar(10)	N

ACADEMICSTAFF		
StaffID	**integer(10)**	
FirstName	varchar(100)	
LastName	varchar(100)	
EmploymentStartDate	date	
PhoneExtension	varchar(20)	
OfficeNumber	varchar(20)	
Gender	varchar(10)	N
Salary	numeric(10, 2)	N
Position	varchar(100)	
HighestQualification	varchar(100)	
DepartmentCode	varchar(10)	
COURSECOORDINATORSStaffID	*integer(10)*	

NEXTOFKIN	
NextOfKinID	**integer(10)**
StudentID	integer(10)
FirstName	varchar(100)
LastName	varchar(100)
Address	varchar(200)
Phone	varchar(20)
Relationship	varchar(50)
STUDENTSStudentID	*integer(10)*

STUDENTS		
StudentID	**integer(10)**	
FirstName	varchar(100)	
LastName	varchar(100)	
Address	varchar(200)	
Town	varchar(100)	
Postcode	varchar(20)	
DOB	date	
Gender	varchar(10)	N
HasStudentLoan	numeric(1, 0)	

STUDENTPERFORMANCE		
StudentID	**integer(10)**	
ProgrammeCode	varchar(10)	N
CourseCode	varchar(10)	N
Grade	varchar(10)	
STUDENTSStudentID	*integer(10)*	

PROGRAMMEDIRECTORS	
StaffID	**integer(10)**
ProgrammeCode	*varchar(10)*

DEPARTMENTHEADS		
DepartmentCode	**varchar(10)**	
StaffID	integer(10)	N
StartDate	date	
DEPARTMENTSDepartmentCode	*varchar(10)*	

DEPARTMENTS	
DepartmentCode	**varchar(10)**
DepartmentName	varchar(200)
DepartmentPhone	varchar(20)
DepartmentLocation	varchar(100)

ASSUMPTIONS

1.Programme Structure:
 * A ProgrammeCode uniquely identifies each programming.
 * Programmes are made up of various courses, and each course can be part of multiple programmes, resulting in a many-to-many relationship governed by the ProgrammeCourses entity.

2.Course Details:
 * Each course is individually recognized by its CourseCode.
 * Courses have properties like name and credits.
 * A course may be taught by many staff members with various coordinators, demonstrating independent teaching and coordinating responsibilities.

3.Staff Roles:
 * Academic faculty are uniquely recognized by their StaffIDs.
 * Staff members can work as both teachers (CourseTeaching) and coordinators (CourseCoordinators).
 * Every employee is assigned to one department.
 * Staff members can also function as program directors and department leaders, demonstrating a hierarchical structure within the organization.

4.Student Information:
 * Each student receives a unique StudentID.
 * The students' personal information is complete, including their address, town, and loan status.
 * Students may have several next of kin contacts, implying a one-to-many link with their next of kin.

5.Performance Tracking:
 * Student performance is monitored for each course within a program using the StudentPerformance object.
 * Each performance record is associated with a specific student, program, and course, implying extensive tracking of academic achievement.

6.Department Structure:
 * A DepartmentCode uniquely identifies departments.
 * Departments feature properties including name, phone number, and location.
 * Each department can have one head at a time, and they are chosen from the academic staff.

7.Teaching and Coordination:
 * Staff teaching load is measured in terms of weekly hours per course (CourseTeaching entity).
 * Course coordination tasks are unique and recorded independently of teaching obligations.

8.Relationships and Integrity:
 * Foreign key restrictions enhance data integrity by connecting relevant entries from distinct entities.
 * The database structure can handle complicated queries on programme composition, teaching assignments, student performance, and staff roles.

9.Programmes and Directors:
 * Each programme has an academic staff member as its director.
 * This connection implies that authorized staff are responsible for administrative control and administration of academic courses.

10. Date Fields:
 * The employment start dates of staff members and department heads show how tenure and position changes have evolved throughout time.

Part B Database Implementation

Task 3: Create tables

```
//1. PROGRAMMES TABLE
CREATE TABLE Programmes (
   ProgrammeCode VARCHAR2(10) PRIMARY KEY,
   Levels VARCHAR2(20) NOT NULL,
   Points NUMBER CHECK (Points > 0),
   Duration NUMBER CHECK (Duration > 0),
   Campus VARCHAR2(100) NOT NULL,
   Semester VARCHAR2(20) NOT NULL
);
```

```
//2. COURSES TABLE
CREATE TABLE Courses (
   CourseCode VARCHAR2(10) PRIMARY KEY,
   CourseName VARCHAR2(200) NOT NULL,
   Credits NUMBER CHECK (Credits > 0)
);
```

```
//3. PROGRAMMECOURSES
CREATE TABLE ProgrammeCourses (
   ProgrammeCode VARCHAR2(10),
   CourseCode VARCHAR2(10),
   PRIMARY KEY (ProgrammeCode, CourseCode),
   FOREIGN KEY (ProgrammeCode) REFERENCES Programmes(ProgrammeCode),
   FOREIGN KEY (CourseCode) REFERENCES Courses(CourseCode)
);
```

```
//4. DEPARTMENTS TABLE
CREATE TABLE Departments (
   DepartmentCode VARCHAR2(10) PRIMARY KEY,
   DepartmentName VARCHAR2(200) NOT NULL,
   DepartmentPhone VARCHAR2(20) NOT NULL,
   DepartmentLocation VARCHAR2(100) NOT NULL
);
```

```
//5. ACADEMICSTAFF TABLE
CREATE TABLE AcademicStaff (
   StaffID NUMBER PRIMARY KEY,
   FirstName VARCHAR2(100) NOT NULL,
   LastName VARCHAR2(100) NOT NULL,
   EmploymentStartDate DATE NOT NULL,
   PhoneExtension VARCHAR2(20) NOT NULL,
   OfficeNumber VARCHAR2(20) NOT NULL,
   Gender VARCHAR2(10) CHECK (Gender IN ('Male', 'Female', 'Other')),
   Salary NUMBER(10,2) CHECK (Salary > 0),
   Position VARCHAR2(100) NOT NULL,
   HighestQualification VARCHAR2(100) NOT NULL,
   DepartmentCode VARCHAR2(10) NOT NULL,
   FOREIGN KEY (DepartmentCode) REFERENCES Departments(DepartmentCode)
);
```

```
//6. DEPARTMENTHEADS TABLE
CREATE TABLE DepartmentHeads (
    DepartmentCode VARCHAR2(10),
    StaffID NUMBER,
    StartDate DATE NOT NULL,
    PRIMARY KEY (DepartmentCode, StaffID),
    FOREIGN KEY (DepartmentCode) REFERENCES Departments(DepartmentCode),
    FOREIGN KEY (StaffID) REFERENCES AcademicStaff(StaffID)
);
```

```
//7. STUDENTS TABLE
CREATE TABLE Students (
    StudentID NUMBER PRIMARY KEY,
    FirstName VARCHAR2(100) NOT NULL,
    LastName VARCHAR2(100) NOT NULL,
    Address VARCHAR2(200) NOT NULL,
    Town VARCHAR2(100) NOT NULL,
    Postcode VARCHAR2(20) NOT NULL,
    DOB DATE NOT NULL,
    Gender VARCHAR2(10) CHECK (Gender IN ('Male', 'Female', 'Other')),
    HasStudentLoan NUMBER(1,0) NOT NULL CHECK (HasStudentLoan IN (0, 1))
);
```

```
//8. NEXTOFKIN TABLE
CREATE TABLE NextOfKin (
    NextOfKinID NUMBER PRIMARY KEY,
    StudentID NUMBER NOT NULL UNIQUE,
    FirstName VARCHAR2(100) NOT NULL,
    LastName VARCHAR2(100) NOT NULL,
    Address VARCHAR2(200) NOT NULL,
    Phone VARCHAR2(20) NOT NULL,
    Relationship VARCHAR2(50) NOT NULL,
    FOREIGN KEY (StudentID) REFERENCES Students(StudentID)
);
```

```
//9. STUDENTPERFORMANCE TABLE
CREATE TABLE StudentPerformance (
   StudentID NUMBER,
   ProgrammeCode VARCHAR2(10),
   CourseCode VARCHAR2(10),
   Grade VARCHAR2(10) NOT NULL,
   PRIMARY KEY (StudentID, ProgrammeCode, CourseCode),
   FOREIGN KEY (StudentID) REFERENCES Students(StudentID),
   FOREIGN KEY (ProgrammeCode) REFERENCES Programmes(ProgrammeCode),
   FOREIGN KEY (CourseCode) REFERENCES Courses(CourseCode)
);
```

```
//10. COURSECOORDINATOR TABLE
CREATE TABLE CourseCoordinators (
   StaffID NUMBER,
   CourseCode VARCHAR2(10),
   PRIMARY KEY (StaffID, CourseCode),
   FOREIGN KEY (StaffID) REFERENCES AcademicStaff(StaffID),
   FOREIGN KEY (CourseCode) REFERENCES Courses(CourseCode)
);
```

```
//11. PROGRAMMEDIRECTORS TABLE
CREATE TABLE ProgrammeDirectors (
   StaffID NUMBER,
   ProgrammeCode VARCHAR2(10),
   PRIMARY KEY (StaffID, ProgrammeCode),
   FOREIGN KEY (StaffID) REFERENCES AcademicStaff(StaffID),
   FOREIGN KEY (ProgrammeCode) REFERENCES Programmes(ProgrammeCode)
);
```

```
//12. COURSETEACHING TABLE
CREATE TABLE CourseTeaching (
   StaffID NUMBER,
   CourseCode VARCHAR2(10),
   TeachingHoursPerWeek NUMBER CHECK (TeachingHoursPerWeek > 0),
   PRIMARY KEY (StaffID, CourseCode),
   FOREIGN KEY (StaffID) REFERENCES AcademicStaff(StaffID),
   FOREIGN KEY (CourseCode) REFERENCES Courses(CourseCode)
);
```

```
//13. ADITIONAL INDEX TO INCREMENT THE PERFORMANCE
CREATE INDEX idx_students_lastname ON Students(LastName);
CREATE INDEX idx_courses_name ON Courses(CourseName);
CREATE INDEX idx_staff_lastname ON AcademicStaff(LastName);
```

Task 4: Populate Data

```
//PROGRAMMES DATA INSERTION
INSERT INTO Programmes (ProgrammeCode, Levels, Points, Duration, Campus, Semester)
VALUES ('BIT', 'Bachelor', 360, 3, 'City Campus', 'Semester 1');
INSERT INTO Programmes (ProgrammeCode, Levels, Points, Duration, Campus, Semester)
VALUES ('BSE', 'Bachelor', 360, 3, 'City Campus', 'Semester 1');
INSERT INTO Programmes (ProgrammeCode, Levels, Points, Duration, Campus, Semester)
VALUES ('BCS', 'Bachelor', 360, 3, 'City Campus', 'Semester 2');
INSERT INTO Programmes (ProgrammeCode, Levels, Points, Duration, Campus, Semester)
VALUES ('BCOM', 'Bachelor', 360, 3, 'Suburban Campus', 'Semester 1');
INSERT INTO Programmes (ProgrammeCode, Levels, Points, Duration, Campus, Semester)
VALUES ('MIS', 'Master', 180, 2, 'City Campus', 'Semester 2');
INSERT INTO Programmes (ProgrammeCode, Levels, Points, Duration, Campus, Semester)
VALUES ('PHD', 'Doctorate', 360, 4, 'City Campus', 'Semester 1');
```

```
//COURSES DATA INSERTION
INSERT INTO Courses (CourseCode, CourseName, Credits)
VALUES ('PROG101', 'Programming I', 15);
INSERT INTO Courses (CourseCode, CourseName, Credits)
VALUES ('DBMS202', 'Database Systems', 15);
INSERT INTO Courses (CourseCode, CourseName, Credits)
VALUES ('NETW301', 'Computer Networks', 15);
INSERT INTO Courses (CourseCode, CourseName, Credits)
VALUES ('ALGO303', 'Algorithms', 15);
INSERT INTO Courses (CourseCode, CourseName, Credits)
VALUES ('PROJ401', 'Final Year Project', 30);
INSERT INTO Courses (CourseCode, CourseName, Credits)
VALUES ('THESIS601', 'PhD Thesis', 120);
```

```
//PROGRAMMECOURSES DATA INSERTION
INSERT INTO ProgrammeCourses (ProgrammeCode, CourseCode)
VALUES ('BIT', 'PROG101');
INSERT INTO ProgrammeCourses (ProgrammeCode, CourseCode)
VALUES ('BIT', 'DBMS202');
INSERT INTO ProgrammeCourses (ProgrammeCode, CourseCode)
VALUES ('BSE', 'PROG101');
INSERT INTO ProgrammeCourses (ProgrammeCode, CourseCode)
VALUES ('BSE', 'DBMS202');
INSERT INTO ProgrammeCourses (ProgrammeCode, CourseCode)
VALUES ('BCS', 'PROG101');
INSERT INTO ProgrammeCourses (ProgrammeCode, CourseCode)
VALUES ('BCS', 'DBMS202');
```

```
INSERT INTO ProgrammeCourses (ProgrammeCode, CourseCode)
VALUES ('BCOM', 'DBMS202');
INSERT INTO ProgrammeCourses (ProgrammeCode, CourseCode)
VALUES ('MIS', 'NETW301');
INSERT INTO ProgrammeCourses (ProgrammeCode, CourseCode)
VALUES ('MIS', 'ALGO303');
INSERT INTO ProgrammeCourses (ProgrammeCode, CourseCode)
VALUES ('PHD', 'THESIS601');
```

```
//STUDENTS DATA INSERTION
INSERT INTO Students (StudentID, FirstName, LastName, Address, Town, Postcode, DOB, Gender,
HasStudentLoan)
VALUES (1001, 'John', 'Smith', '123 Main St', 'Whangarei', '0112', TO_DATE('1995-03-15', 'YYYY-MM-DD'),
'Male', 1);
INSERT INTO Students (StudentID, FirstName, LastName, Address, Town, Postcode, DOB, Gender,
HasStudentLoan)
VALUES (1002, 'Emma', 'Johnson', '456 Oak Rd', 'Whangarei', '0110', TO_DATE('1998-07-22', 'YYYY-MM-
DD'), 'Female', 0);
INSERT INTO Students (StudentID, FirstName, LastName, Address, Town, Postcode, DOB, Gender,
HasStudentLoan)
VALUES (1003, 'Michael', 'Williams', '789 Elm St', 'Kaikohe', '0471', TO_DATE('1997-11-03', 'YYYY-MM-DD'),
'Male', 1);
INSERT INTO Students (StudentID, FirstName, LastName, Address, Town, Postcode, DOB, Gender,
HasStudentLoan)
VALUES (1004, 'Sophia', 'Brown', '321 Pine Ave', 'Kerikeri', '0230', TO_DATE('1996-06-28', 'YYYY-MM-DD'),
'Female', 0);
INSERT INTO Students (StudentID, FirstName, LastName, Address, Town, Postcode, DOB, Gender,
HasStudentLoan)
VALUES (1005, 'Daniel', 'Jones', '159 Cedar Ln', 'Whangarei', '0112', TO_DATE('1994-02-10', 'YYYY-MM-DD'),
'Male', 1);
INSERT INTO Students (StudentID, FirstName, LastName, Address, Town, Postcode, DOB, Gender,
HasStudentLoan)
VALUES (1006, 'Olivia', 'Garcia', '753 Maple Dr', 'Whangarei', '0110', TO_DATE('1999-09-18', 'YYYY-MM-
DD'), 'Female', 1);
```

```
//NEXTOFKIN DATA INSERTION
INSERT INTO NextOfKin (NextOfKinID, StudentID, FirstName, LastName, Address, Phone, Relationship)
VALUES (2001, 1001, 'Jane', 'Smith', '123 Main St, Whangarei', '09-1234567', 'Mother');
INSERT INTO NextOfKin (NextOfKinID, StudentID, FirstName, LastName, Address, Phone, Relationship)
VALUES (2002, 1002, 'Robert', 'Johnson', '456 Oak Rd, Whangarei', '09-8765432', 'Father');
INSERT INTO NextOfKin (NextOfKinID, StudentID, FirstName, LastName, Address, Phone, Relationship)
VALUES (2003, 1003, 'Emily', 'Williams', '789 Elm St, Kaikohe', '09-2468013', 'Sister');
INSERT INTO NextOfKin (NextOfKinID, StudentID, FirstName, LastName, Address, Phone, Relationship)
VALUES (2004, 1004, 'David', 'Brown', '321 Pine Ave, Kerikeri', '09-5679012', 'Brother');
INSERT INTO NextOfKin (NextOfKinID, StudentID, FirstName, LastName, Address, Phone, Relationship)
```

```
VALUES (2005, 1005, 'Sarah', 'Jones', '159 Cedar Ln, Whangarei', '09-3456789', 'Mother');
```

```
//DEPARTMENTS DATA INSERTION
INSERT INTO Departments (DepartmentCode, DepartmentName, DepartmentPhone, DepartmentLocation)
VALUES ('COMP', 'Department of Computer Science', '09-1234567', 'Z Building');
INSERT INTO Departments (DepartmentCode, DepartmentName, DepartmentPhone, DepartmentLocation)
VALUES ('INFO', 'Department of Information Systems', '09-2345678', 'X Building');
INSERT INTO Departments (DepartmentCode, DepartmentName, DepartmentPhone, DepartmentLocation)
VALUES ('SOFT', 'Department of Software Engineering', '09-3456789', 'Y Building');
INSERT INTO Departments (DepartmentCode, DepartmentName, DepartmentPhone, DepartmentLocation)
VALUES ('BUSI', 'Department of Business', '09-4567890', 'W Building');
INSERT INTO Departments (DepartmentCode, DepartmentName, DepartmentPhone, DepartmentLocation)
VALUES ('MATH', 'Department of Mathematics', '09-5678901', 'V Building');
INSERT INTO Departments (DepartmentCode, DepartmentName, DepartmentPhone, DepartmentLocation)
VALUES ('ENGL', 'Department of English', '09-6789012', 'A Building');
INSERT INTO Departments (DepartmentCode, DepartmentName, DepartmentPhone, DepartmentLocation)
VALUES ('HIST', 'Department of History', '09-7890123', 'B Building');
INSERT INTO Departments (DepartmentCode, DepartmentName, DepartmentPhone, DepartmentLocation)
VALUES ('CHEM', 'Department of Chemistry', '09-8901234', 'C Building');
INSERT INTO Departments (DepartmentCode, DepartmentName, DepartmentPhone, DepartmentLocation)
VALUES ('PHYS', 'Department of Physics', '09-9012345', 'D Building');
INSERT INTO Departments (DepartmentCode, DepartmentName, DepartmentPhone, DepartmentLocation)
VALUES ('BIOL', 'Department of Biology', '09-0123456', 'E Building');
```

```
//ACADEMICSTAFF DATA INSERTION
INSERT INTO AcademicStaff (StaffID, FirstName, LastName, EmploymentStartDate, PhoneExtension,
OfficeNumber, Gender, Salary, Position, HighestQualification, DepartmentCode)
VALUES (1001, 'Manuel', 'Santos', TO_DATE('2010-01-01', 'YYYY-MM-DD'), '1001', 'Z101', 'Male', 80000.00,
'Professor', 'PhD', 'COMP');
INSERT INTO AcademicStaff (StaffID, FirstName, LastName, EmploymentStartDate, PhoneExtension,
OfficeNumber, Gender, Salary, Position, HighestQualification, DepartmentCode)
VALUES (1002, 'Sarah', 'Vidal', TO_DATE('2015-06-15', 'YYYY-MM-DD'), '1002', 'X201', 'Female', 65000.00,
'Senior Lecturer', 'Master', 'INFO');
INSERT INTO AcademicStaff (StaffID, FirstName, LastName, EmploymentStartDate, PhoneExtension,
OfficeNumber, Gender, Salary, Position, HighestQualification, DepartmentCode)
VALUES (1003, 'Jorge', 'Guillen', TO_DATE('2012-09-01', 'YYYY-MM-DD'), '1003', 'Y102', 'Male', 70000.00,
'Lecturer', 'PhD', 'SOFT');
INSERT INTO AcademicStaff (StaffID, FirstName, LastName, EmploymentStartDate, PhoneExtension,
OfficeNumber, Gender, Salary, Position, HighestQualification, DepartmentCode)
VALUES (1004, 'Sofia', 'Suarez', TO_DATE('2018-03-01', 'YYYY-MM-DD'), '1004', 'W301', 'Female', 55000.00,
'Lecturer', 'Master', 'BUSI');
INSERT INTO AcademicStaff (StaffID, FirstName, LastName, EmploymentStartDate, PhoneExtension,
OfficeNumber, Gender, Salary, Position, HighestQualification, DepartmentCode)
VALUES (1005, 'Drielly', 'Velasco', TO_DATE('2020-07-01', 'YYYY-MM-DD'), '1005', 'V101', 'Male', 60000.00,
'Senior Lecturer', 'PhD', 'MATH');
```

```
//DEPARTMENTHEADS DATA INSERTION
INSERT INTO DepartmentHeads (DepartmentCode, StaffID, StartDate)
VALUES ('COMP', 1001, TO_DATE('2015-01-01', 'YYYY-MM-DD'));
INSERT INTO DepartmentHeads (DepartmentCode, StaffID, StartDate)
VALUES ('INFO', 1002, TO_DATE('2018-06-01', 'YYYY-MM-DD'));
INSERT INTO DepartmentHeads (DepartmentCode, StaffID, StartDate)
VALUES ('SOFT', 1003, TO_DATE('2017-02-15', 'YYYY-MM-DD'));
INSERT INTO DepartmentHeads (DepartmentCode, StaffID, StartDate)
VALUES ('BUSI', 1004, TO_DATE('2020-03-01', 'YYYY-MM-DD'));
INSERT INTO DepartmentHeads (DepartmentCode, StaffID, StartDate)
VALUES ('MATH', 1005, TO_DATE('2022-09-01', 'YYYY-MM-DD'));
```

```
//PROGRAMMEDIRECTORS DATA INSERTION
INSERT INTO ProgrammeDirectors (StaffID, ProgrammeCode)
VALUES (1001, 'BIT');
INSERT INTO ProgrammeDirectors (StaffID, ProgrammeCode)
VALUES (1002, 'MIS');
INSERT INTO ProgrammeDirectors (StaffID, ProgrammeCode)
VALUES (1003, 'BSE');
INSERT INTO ProgrammeDirectors (StaffID, ProgrammeCode)
VALUES (1004, 'BCOM');
INSERT INTO ProgrammeDirectors (StaffID, ProgrammeCode)
VALUES (1005, 'BCS');
```

```
//COURSECOORDINATORS DATA INSERTION
INSERT INTO CourseCoordinators (StaffID, CourseCode)
VALUES (1001, 'PROG101');
INSERT INTO CourseCoordinators (StaffID, CourseCode)
VALUES (1002, 'DBMS202');
INSERT INTO CourseCoordinators (StaffID, CourseCode)
VALUES (1003, 'NETW301');
INSERT INTO CourseCoordinators (StaffID, CourseCode)
VALUES (1004, 'ALGO303');
INSERT INTO CourseCoordinators (StaffID, CourseCode)
VALUES (1005, 'PROJ401');
```

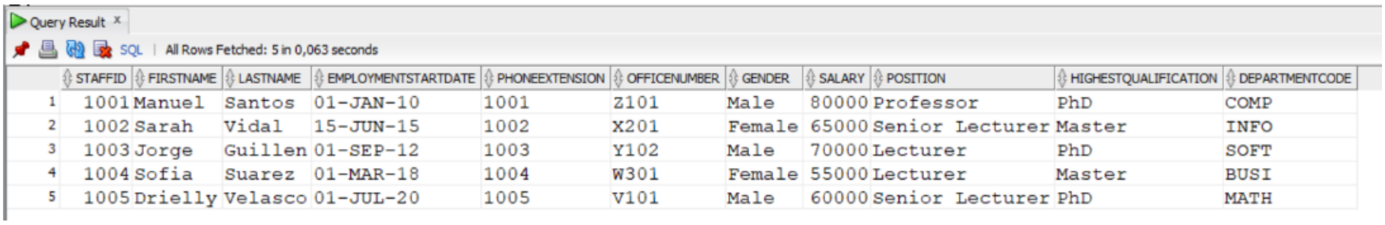
```
//COURSETEACHING DATA INSERTION
INSERT INTO CourseTeaching (StaffID, CourseCode, TeachingHoursPerWeek)
VALUES (1001, 'PROG101', 4);
INSERT INTO CourseTeaching (StaffID, CourseCode, TeachingHoursPerWeek)
VALUES (1002, 'DBMS202', 3);
INSERT INTO CourseTeaching (StaffID, CourseCode, TeachingHoursPerWeek)
VALUES (1003, 'NETW301', 5);
INSERT INTO CourseTeaching (StaffID, CourseCode, TeachingHoursPerWeek)
```

```
VALUES (1004, 'ALGO303', 4);
INSERT INTO CourseTeaching (StaffID, CourseCode, TeachingHoursPerWeek)
VALUES (1005, 'PROJ401', 2);
```

```
//STUDENTPERFORMANCE DATA INSERTION
INSERT INTO StudentPerformance (StudentID, ProgrammeCode, CourseCode, Grade)
VALUES (1001, 'BIT', 'PROG101', 'A');
INSERT INTO StudentPerformance (StudentID, ProgrammeCode, CourseCode, Grade)
VALUES (1002, 'MIS', 'DBMS202', 'B+');
INSERT INTO StudentPerformance (StudentID, ProgrammeCode, CourseCode, Grade)
VALUES (1003, 'BSE', 'PROG101', 'A-');
INSERT INTO StudentPerformance (StudentID, ProgrammeCode, CourseCode, Grade)
VALUES (1004, 'BCOM', 'DBMS202', 'B');
INSERT INTO StudentPerformance (StudentID, ProgrammeCode, CourseCode, Grade)
VALUES (1005, 'BCS', 'PROG101', 'A');
```

Task 4: Evidence of Sample data – "SELECT* FROM *table_name*"

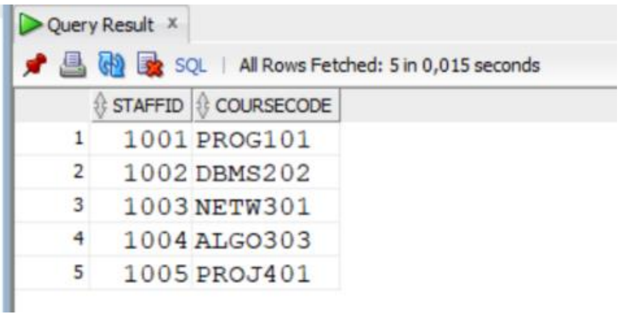
```
SELECT* FROM ACADEMICSTAFF;
```



All Rows Fetched: 5 in 0,063 seconds

	STAFFID	FIRSTNAME	LASTNAME	EMPLOYMENTSTARTDATE	PHONEEXTENSION	OFFICENUMBER	GENDER	SALARY	POSITION	HIGHESTQUALIFICATION	DEPARTMENTCODE
1	1001	Manuel	Santos	01-JAN-10	1001	Z101	Male	80000	Professor	PhD	COMP
2	1002	Sarah	Vidal	15-JUN-15	1002	X201	Female	65000	Senior Lecturer	Master	INFO
3	1003	Jorge	Guillen	01-SEP-12	1003	Y102	Male	70000	Lecturer	PhD	SOFT
4	1004	Sofia	Suarez	01-MAR-18	1004	W301	Female	55000	Lecturer	Master	BUSI
5	1005	Drielly	Velasco	01-JUL-20	1005	V101	Male	60000	Senior Lecturer	PhD	MATH

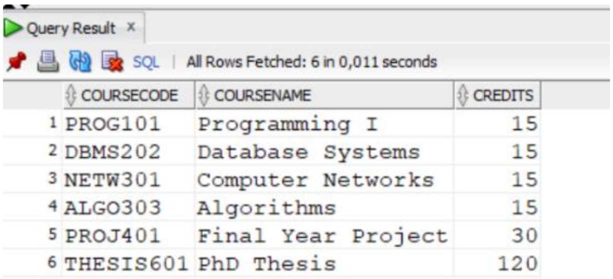
```
SELECT* FROM COURSECOORDINATORS;
```



All Rows Fetched: 5 in 0,015 seconds

	STAFFID	COURSECODE
1	1001	PROG101
2	1002	DBMS202
3	1003	NETW301
4	1004	ALGO303
5	1005	PROJ401

```
SELECT* FROM COURSES;
```

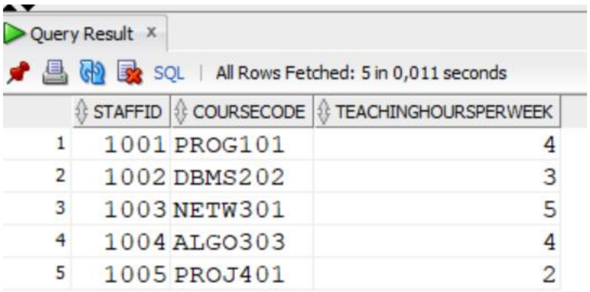


Query Result

All Rows Fetched: 6 in 0,011 seconds

	COURSECODE	COURSENAME	CREDITS
1	PROG101	Programming I	15
2	DBMS202	Database Systems	15
3	NETW301	Computer Networks	15
4	ALGO303	Algorithms	15
5	PROJ401	Final Year Project	30
6	THESIS601	PhD Thesis	120

SELECT* FROM COURSETEACHING;

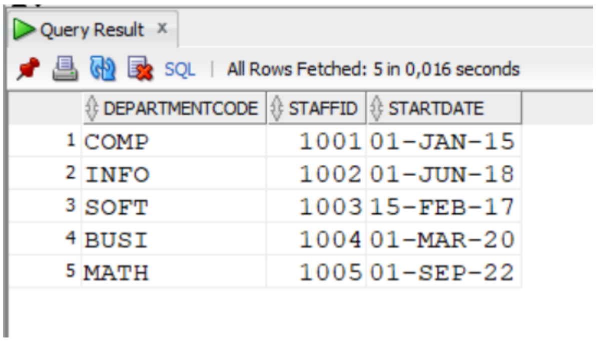


Query Result

All Rows Fetched: 5 in 0,011 seconds

	STAFFID	COURSECODE	TEACHINGHOURSPERWEEK
1	1001	PROG101	4
2	1002	DBMS202	3
3	1003	NETW301	5
4	1004	ALGO303	4
5	1005	PROJ401	2

SELECT* FROM DEPARTMENTHEADS;

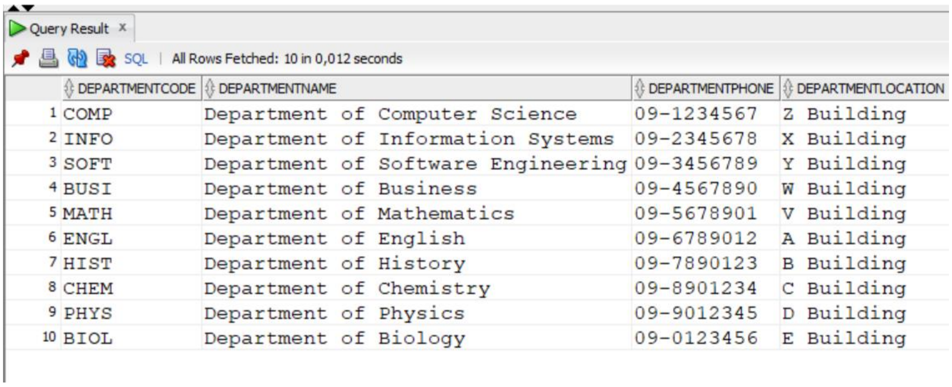


Query Result

All Rows Fetched: 5 in 0,016 seconds

	DEPARTMENTCODE	STAFFID	STARTDATE
1	COMP	1001	01-JAN-15
2	INFO	1002	01-JUN-18
3	SOFT	1003	15-FEB-17
4	BUSI	1004	01-MAR-20
5	MATH	1005	01-SEP-22

SELECT* FROM DEPARTMENTS;

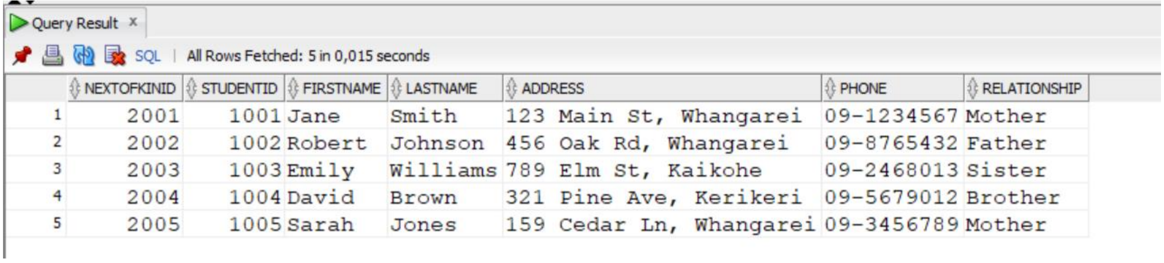


Query Result

All Rows Fetched: 10 in 0,012 seconds

	DEPARTMENTCODE	DEPARTMENTNAME	DEPARTMENTPHONE	DEPARTMENTLOCATION
1	COMP	Department of Computer Science	09-1234567	Z Building
2	INFO	Department of Information Systems	09-2345678	X Building
3	SOFT	Department of Software Engineering	09-3456789	Y Building
4	BUSI	Department of Business	09-4567890	W Building
5	MATH	Department of Mathematics	09-5678901	V Building
6	ENGL	Department of English	09-6789012	A Building
7	HIST	Department of History	09-7890123	B Building
8	CHEM	Department of Chemistry	09-8901234	C Building
9	PHYS	Department of Physics	09-9012345	D Building
10	BIOL	Department of Biology	09-0123456	E Building

SELECT* FROM NEXTOFKIN;

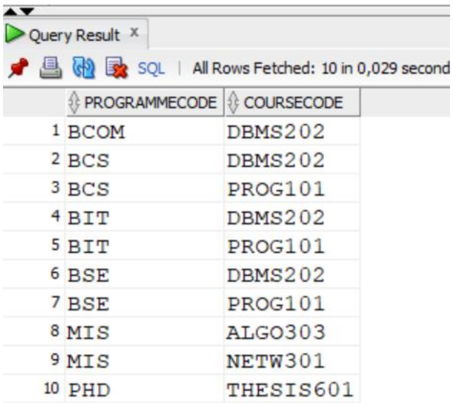


Query Result ×

SQL | All Rows Fetched: 5 in 0,015 seconds

	NEXTOFKINID	STUDENTID	FIRSTNAME	LASTNAME	ADDRESS	PHONE	RELATIONSHIP
1	2001	1001	Jane	Smith	123 Main St, Whangarei	09-1234567	Mother
2	2002	1002	Robert	Johnson	456 Oak Rd, Whangarei	09-8765432	Father
3	2003	1003	Emily	Williams	789 Elm St, Kaikohe	09-2468013	Sister
4	2004	1004	David	Brown	321 Pine Ave, Kerikeri	09-5679012	Brother
5	2005	1005	Sarah	Jones	159 Cedar Ln, Whangarei	09-3456789	Mother

SELECT* FROM PROGRAMMECOURSES;

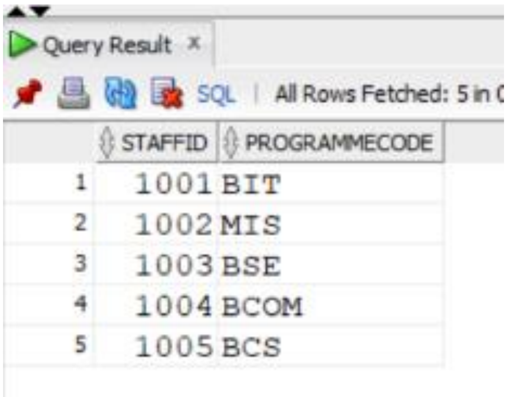


Query Result ×

SQL | All Rows Fetched: 10 in 0,029 second

	PROGRAMMECODE	COURSECODE
1	BCOM	DBMS202
2	BCS	DBMS202
3	BCS	PROG101
4	BIT	DBMS202
5	BIT	PROG101
6	BSE	DBMS202
7	BSE	PROG101
8	MIS	ALGO303
9	MIS	NETW301
10	PHD	THESIS601

SELECT* FROM PROGRAMMEDIRECTORS;

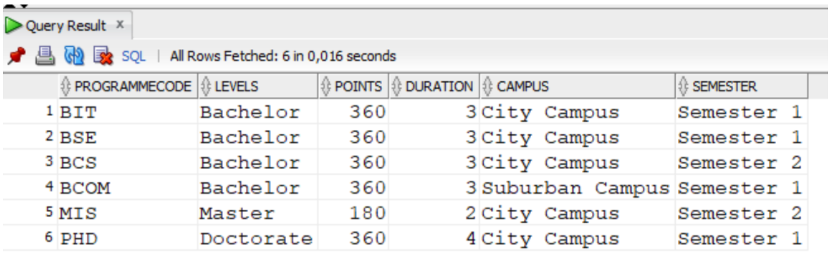


Query Result ×

SQL | All Rows Fetched: 5 in (

	STAFFID	PROGRAMMECODE
1	1001	BIT
2	1002	MIS
3	1003	BSE
4	1004	BCOM
5	1005	BCS

SELECT* FROM PROGRAMMES;

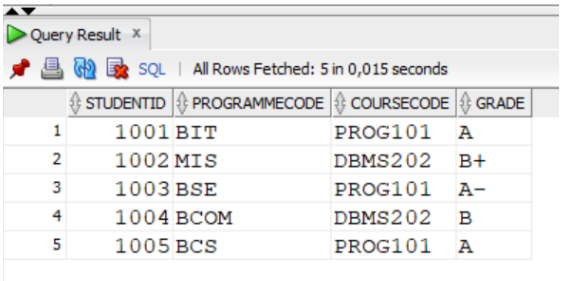


Query Result ×

SQL | All Rows Fetched: 6 in 0,016 seconds

	PROGRAMMECODE	LEVELS	POINTS	DURATION	CAMPUS	SEMESTER
1	BIT	Bachelor	360	3	City Campus	Semester 1
2	BSE	Bachelor	360	3	City Campus	Semester 1
3	BCS	Bachelor	360	3	City Campus	Semester 2
4	BCOM	Bachelor	360	3	Suburban Campus	Semester 1
5	MIS	Master	180	2	City Campus	Semester 2
6	PHD	Doctorate	360	4	City Campus	Semester 1

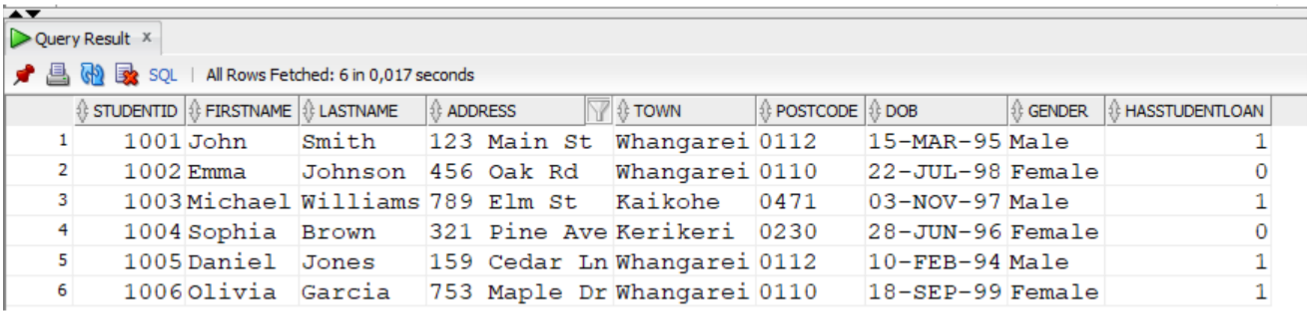
SELECT* FROM STUDENTPERFORMANCE;



All Rows Fetched: 5 in 0,015 seconds

	STUDENTID	PROGRAMMECODE	COURSECODE	GRADE
1	1001	BIT	PROG101	A
2	1002	MIS	DBMS202	B+
3	1003	BSE	PROG101	A-
4	1004	BCOM	DBMS202	B
5	1005	BCS	PROG101	A

SELECT* FROM STUDENTS;



All Rows Fetched: 6 in 0,017 seconds

	STUDENTID	FIRSTNAME	LASTNAME	ADDRESS	TOWN	POSTCODE	DOB	GENDER	HASSTUDENTLOAN
1	1001	John	Smith	123 Main St	Whangarei	0112	15-MAR-95	Male	1
2	1002	Emma	Johnson	456 Oak Rd	Whangarei	0110	22-JUL-98	Female	0
3	1003	Michael	Williams	789 Elm St	Kaikohe	0471	03-NOV-97	Male	1
4	1004	Sophia	Brown	321 Pine Ave	Kerikeri	0230	28-JUN-96	Female	0
5	1005	Daniel	Jones	159 Cedar Ln	Whangarei	0112	10-FEB-94	Male	1
6	1006	Olivia	Garcia	753 Maple Dr	Whangarei	0110	18-SEP-99	Female	1

Part C Construct SQL Queries

Task 5: **Construct five SQL Queries**

Query 1

Purpose of this query

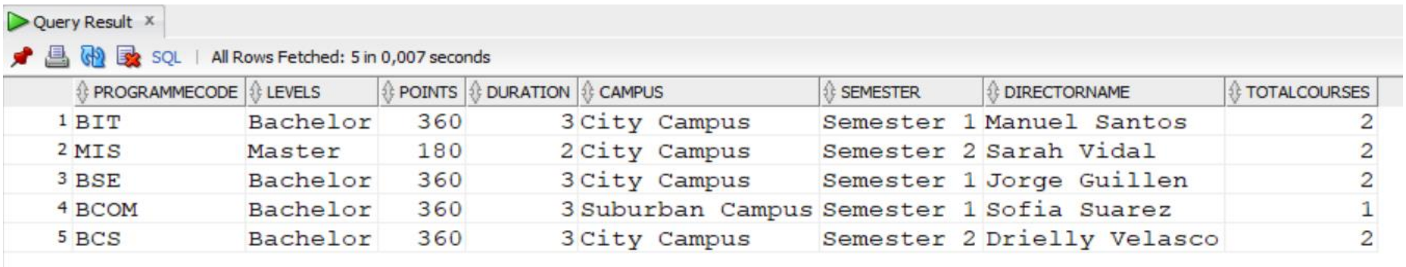
The query's objective is to obtain a list of all academic programs, as well as the complete name of the program director and the total number of courses included in each program.

SQL Query

```
SELECT p.ProgrammeCode, p.Levels, p.Points, p.Duration, p.Campus, p.Semester,

a.FirstName || ' ' || a.LastName AS DirectorName,

(SELECT COUNT(*) FROM ProgrammeCourses pc WHERE pc.ProgrammeCode = p.ProgrammeCode) AS TotalCourses

FROM Programmes p

JOIN ProgrammeDirectors pd ON p.ProgrammeCode = pd.ProgrammeCode
```

JOIN AcademicStaff a ON pd.StaffID = a.StaffID;

Result/Output of this Query



Query Result ×

SQL | All Rows Fetched: 5 in 0,007 seconds

	PROGRAMMECODE	LEVELS	POINTS	DURATION	CAMPUS	SEMESTER	DIRECTORNAME	TOTALCOURSES
1	BIT	Bachelor	360	3	City Campus	Semester 1	Manuel Santos	2
2	MIS	Master	180	2	City Campus	Semester 2	Sarah Vidal	2
3	BSE	Bachelor	360	3	City Campus	Semester 1	Jorge Guillen	2
4	BCOM	Bachelor	360	3	Suburban Campus	Semester 1	Sofia Suarez	1
5	BCS	Bachelor	360	3	City Campus	Semester 2	Drielly Velasco	2

Query 2

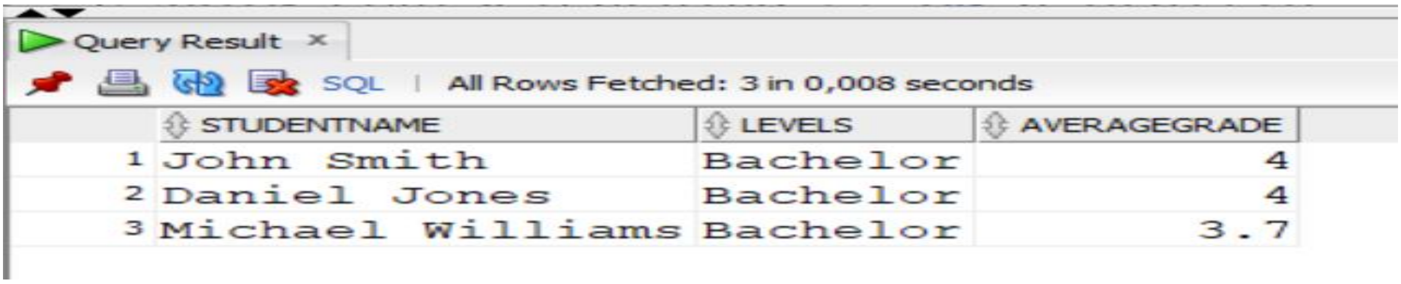
Purpose of this query

The query's purpose is to obtain the full names of students having student loans, as well as the name of the academic program in which they are enrolled and the average of their grades in that program.

SQL Query

```
SELECT s.FirstName || ' ' || s.LastName AS StudentName, p.Levels, AVG(CASE
WHEN sp.Grade = 'A' THEN 4
WHEN sp.Grade = 'B+' THEN 3.5
WHEN sp.Grade = 'B' THEN 3
WHEN sp.Grade = 'A-' THEN 3.7
ELSE 0
END) AS AverageGrade
FROM Students s
JOIN StudentPerformance sp ON s.StudentID = sp.StudentID
JOIN Programmes p ON sp.ProgrammeCode = p.ProgrammeCode
WHERE s.HasStudentLoan = 1
GROUP BY s.FirstName, s.LastName, p.Levels;
```

Result/Output of this Query



Query Result ×

SQL | All Rows Fetched: 3 in 0,008 seconds

	STUDENTNAME	LEVELS	AVERAGEGRADE
1	John Smith	Bachelor	4
2	Daniel Jones	Bachelor	4
3	Michael Williams	Bachelor	3.7

Query 3

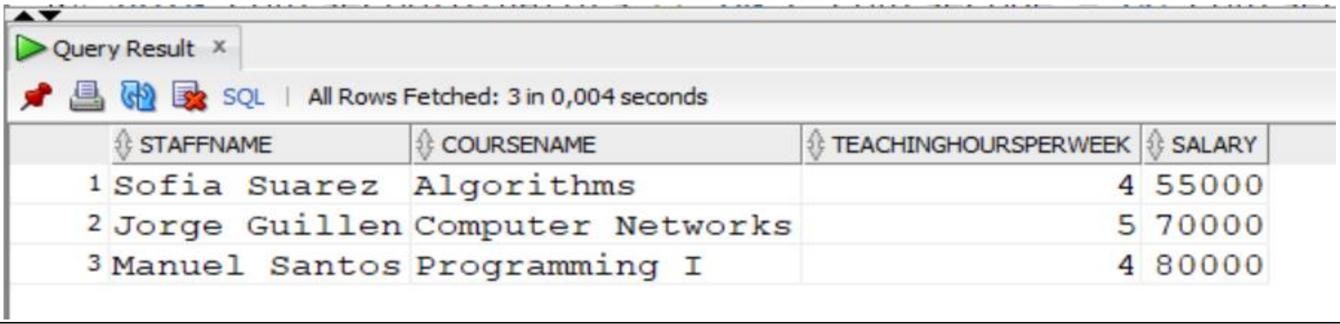
Purpose of this query

The query's objective is to obtain the full names of academic staff members who teach more than four weekly hours in any course, as well as the course name and teaching hours.

SQL Query

```
SELECT a.FirstName || ' ' || a.LastName AS StaffName, c.CourseName, ct.TeachingHoursPerWeek, a.Salary
FROM AcademicStaff a
JOIN CourseTeaching ct ON a.StaffID = ct.StaffID
JOIN Courses c ON ct.CourseCode = c.CourseCode
WHERE ct.TeachingHoursPerWeek > 3 AND a.Salary BETWEEN 50000 AND 90000;
```

Result/Output of this Query



All Rows Fetched: 3 in 0,004 seconds

	STAFFNAME	COURSENAME	TEACHINGHOURSPERWEEK	SALARY
1	Sofia Suarez	Algorithms	4	55000
2	Jorge Guillen	Computer Networks	5	70000
3	Manuel Santos	Programming I	4	80000

Query 4

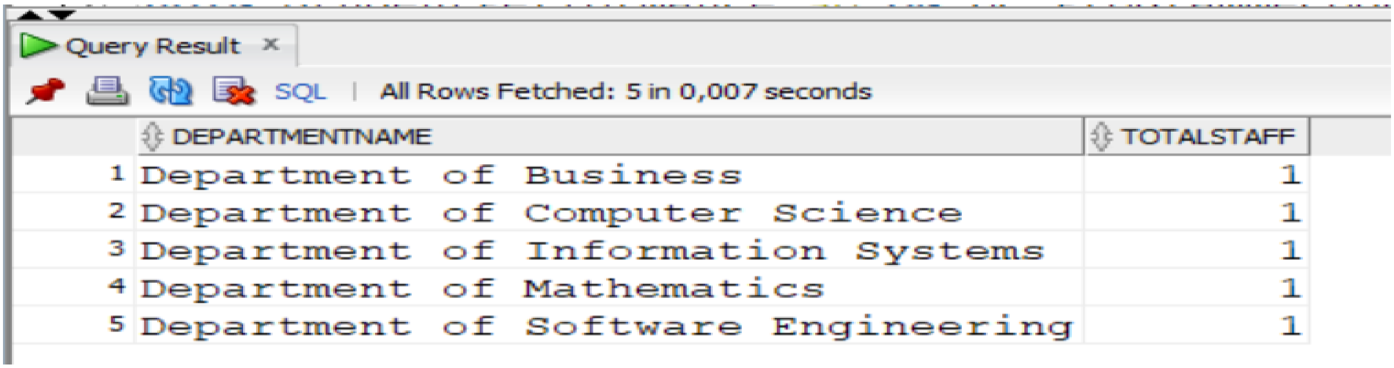
Purpose of this query

The query's purpose is to obtain the department name and total number of academic staff members for departments with at least one appointed department head.

SQL Query

```
SELECT d.DepartmentName, COUNT(a.StaffID) AS TotalStaff
FROM Departments d
LEFT JOIN AcademicStaff a ON d.DepartmentCode = a.DepartmentCode
JOIN DepartmentHeads dh ON d.DepartmentCode = dh.DepartmentCode
GROUP BY d.DepartmentName;
```

Result/Output of this Query



All Rows Fetched: 5 in 0,007 seconds

	DEPARTMENTNAME	TOTALSTAFF
1	Department of Business	1
2	Department of Computer Science	1
3	Department of Information Systems	1
4	Department of Mathematics	1
5	Department of Software Engineering	1

Query 5

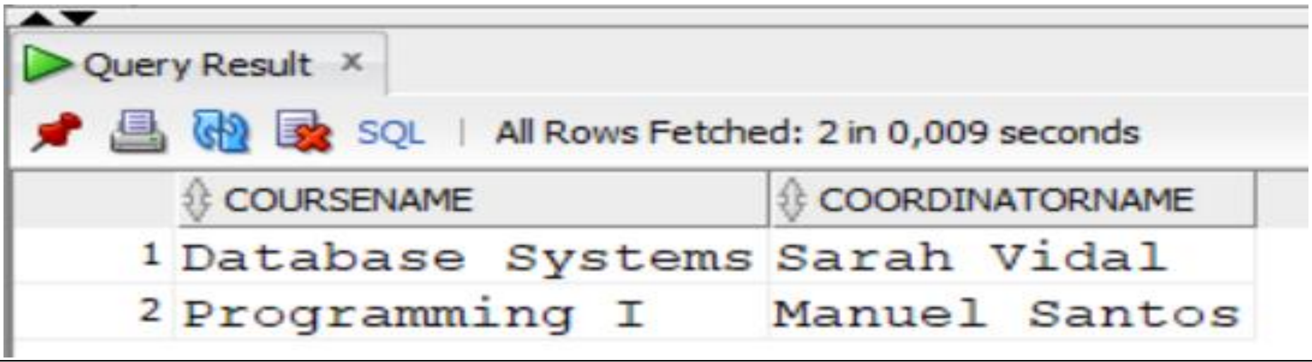
Purpose of this query

The query's objective is to retrieve the course name and the complete name of the course coordinator, in that order, for courses with an appointed coordinator and at least one registered student.

SQL Query

```
SELECT c.CourseName, a.FirstName || ' ' || a.LastName AS CoordinatorName
FROM Courses c
JOIN CourseCoordinators cc ON c.CourseCode = cc.CourseCode
JOIN AcademicStaff a ON cc.StaffID = a.StaffID
JOIN ProgrammeCourses pc ON c.CourseCode = pc.CourseCode
JOIN StudentPerformance sp ON pc.ProgrammeCode = sp.ProgrammeCode AND pc.CourseCode =
sp.CourseCode
GROUP BY c.CourseName, a.FirstName, a.LastName
ORDER BY c.CourseName;
```

Result/Output of this Query



All Rows Fetched: 2 in 0,009 seconds

	COURSENAME	COORDINATORNAME
1	Database Systems	Sarah Vidal
2	Programming I	Manuel Santos