```
USE Student2550;

/*  1. Create a query that returns the average cost for all courses. (Round to two
places, and
alias the column heading).
(1 Row)  */

SELECT
    ROUND(AVG(Cost), 2) AS 'Average Cost'
FROM
    Course;

/*  2. Create a query that returns the total number of unique Students that
registered during February 2007. Alias
the column as "February 2007 Registrations".
(1 Row)  */

SELECT DISTINCT
    COUNT(Registration_Date) AS 'February 2007 Registrations'
FROM
    Student
WHERE
    MONTH(Registration_Date) = 2
        AND YEAR(Registration_Date) = 2007;

/*  3. Create a query that returns the average, highest and lowest final exam
scores for
Section 147. Display the average exam score with 2 decimal places.
(1 Row)  */

SELECT
    ROUND(AVG(Numeric_Grade), 2) AS 'Average Final Exam Score',
    MAX(Numeric_Grade) AS 'Highest Final Exam Score',
    MIN(Numeric_Grade) AS 'Lowest Final Exam Score',
    Grade_Type_Code,
    Section_ID
FROM
    Grade
WHERE
    Grade_Type_Code = 'FI'
        AND Section_ID = 147;

/*  4. List the city, state and “number of zip codes” (alias) for all cities with
more than two
zip codes. Arrange by state and city.
(10 Rows)  */

SELECT
    City, State, COUNT(Zip) AS 'Number of Zip Codes'
FROM
    ZipCode
GROUP BY State , City
HAVING COUNT(Zip) > 2
ORDER BY State , City;

/*  5. Provide a list of Sections and the number of students enrolled in each
section for
students who enrolled on 2/21/2019. Sort from highest to lowest on the number of
students enrolled.
```

```
(14 Rows)  */

SELECT
    COUNT(Student_id) AS 'Number of Students Enrolled',
    Section_id
FROM
    Enrollment
WHERE
    Enroll_Date = '2019-02-21'
GROUP BY section_id
ORDER BY COUNT(Student_id) DESC;

/*  6. Create a query listing the student ID and Average Grade for all students in
Section
86. Sort your list on the student ID and display all of the average grades with 2
decimal places.
(6 Rows)  */

SELECT
    g.Student_Id,
    ROUND(AVG(g.Numeric_Grade), 2) AS 'Average Grade'
FROM
    Grade g
        JOIN
    Section s ON g.Section_ID = s.Section_ID
WHERE
    s.Section_Id = 86
GROUP BY g.Student_ID
ORDER BY g.Student_ID;

/*  7. Create a query to determine the number of sections in which student ID 250
is
enrolled. Your output should contain the student ID and the number of sections
(alias) enrolled.
(1 Row)  */

SELECT
    Student_ID, COUNT(Section_id) AS 'Number of Sections'
FROM
    Enrollment
WHERE
    Student_ID = 250;

/*  8. List the section ID and lowest quiz score (GRADE_TYPE_CODE='QZ') for all
sections
where the low score is greater than a B (greater than 80). Arrange by section id.
(16 rows)  */

SELECT
    Section_id, MIN(Numeric_Grade) AS 'Lowest Quiz Score'
FROM
    Grade
WHERE
    Grade_Type_Code = 'QZ'
GROUP BY Section_ID
HAVING MIN(Numeric_Grade) > 80
ORDER BY Section_ID;

/*  9. List the names of Employers having more than 5 student employees. Your
```

```
output
should contain the employer name and the number of student employees. Arrange
the output on the number of employees from lowest to highest.
(4 Rows)  */

SELECT
    Employer, COUNT(Student_ID) AS 'Number of Student Employees'
FROM
    Student
GROUP BY Employer
HAVING COUNT(Student_ID) > 5
ORDER BY COUNT(Student_ID);

/*  10. List the section ID, number of participation grades (GRADE_TYPE_CODE='PA')
and
lowest participation grade for all sections with more than 15 participation grades.
Arrange by section id.
(6 Rows)  */

SELECT
    Section_ID,
    COUNT(Grade_Type_Code) AS 'Number of Participation Grades',
    MIN(Numeric_Grade) AS 'Lowest Participation Grade'
FROM
    Grade
WHERE
    Grade_Type_Code = 'PA'
GROUP BY Section_ID
HAVING COUNT(Grade_Type_Code) > 15
ORDER BY Section_ID;

/*  11. List the first and last name and phone number for students that live in
Newark, NJ.
Sort on last name and first name.
(3 Rows)  */

SELECT
    s.First_Name, s.Last_Name, s.Phone
FROM
    Student s
        JOIN
    ZipCode zc ON zc.Zip = s.Zip
WHERE
    zc.City = 'Newark' AND zc.State = 'NJ'
ORDER BY s.Last_Name , s.First_Name;

/*  12. (Must use UNION) List all students and instructors last name and zip. Only
list the student and instructor
if they have a zip code. Arrange the list in order of zip then last name.
(277 Rows)  */

SELECT
    Last_Name, Zip
FROM
    Student
WHERE
    Zip IS NOT NULL
UNION SELECT
    Last_Name, Zip
```

```
FROM
    Instructor
WHERE
    Zip IS NOT NULL
ORDER BY Zip , Last_Name;

/*  13. List the course number and description for all 100-level courses taught by
Charles
Lowry. Arrange the list in order of course number.
(6 rows)  */

SELECT
    c.Course_No, Description
FROM
    Course c
        JOIN
    Section s ON c.Course_No = s.Course_No
        JOIN
    Instructor i ON i.Instructor_Id = s.Instructor_Id
WHERE
    c.Course_No LIKE '1%'
        AND i.Last_Name = 'Lowry'
        AND i.First_Name = 'Charles'
ORDER BY Course_no;

/*  14. List the grade type code, grade type description and number of grades for
each grade type in all sections of course
142. Arrange by description.
(5 Rows)  */

SELECT
    gt.Grade_Type_Code,
    gt.Description,
    COUNT(g.Numeric_Grade) AS 'Number of Grades'
FROM
    Grade_Type gt
        JOIN
    Grade g ON gt.Grade_Type_Code = g.Grade_Type_Code
        JOIN
    Section s ON g.Section_Id = s.Section_Id
WHERE
    s.Course_No = 142
GROUP BY gt.Grade_Type_Code
ORDER BY gt.Description;

/*  15. Provide an alphabetic list of students (by the Students’ Full Name – with
last name
first) who have an overall grade average of 93 or higher. The results should be
sorted on Students’ Full name (showing last name first).
(3 Rows)  */

SELECT
    CONCAT(Last_Name, ', ', First_Name) AS 'Full Name'
FROM
    Student
WHERE
    (SELECT
            AVG(Numeric_Grade)
        FROM
```

```
            Grade
        WHERE
            Student_Id = student.Student_Id) >= 93
ORDER BY 'Full Name';

/*  16. List the names and address (including city and state) for all faculty who
have taught
less than 10 course sections.
(2 Rows)  */

SELECT
    i.First_Name,
    i.Last_Name,
    i.Street_Address,
    i.Zip,
    zc.City,
    zc.State,
    COUNT(Section_No) AS 'Sections Taught'
FROM
    Instructor i
        JOIN
    ZipCode zc ON i.Zip = zc.Zip
        JOIN
    Section s ON s.Instructor_id = i.Instructor_id
GROUP BY i.Instructor_ID
HAVING COUNT(Section_No) < 10;

/*  17. List the course number and number of students enrolled in courses that do
not have
a prerequisite. Sort the list by number of students enrolled from highest to
lowest.
(4 Rows)  */

SELECT
    c.Course_No,
    COUNT(e.Student_ID) AS 'Number of Students Enrolled'
FROM
    Course c
        JOIN
    Section s ON c.Course_No = s.Course_No
        JOIN
    Enrollment e ON s.Section_Id = e.Section_Id
WHERE
    c.Prerequisite IS NULL
GROUP BY c.Course_No
ORDER BY 'Number of Students Enrolled';

/*  18. Provide an alphabetic list of students (first and last names) their
enrollment date(s),
and count of their enrollment(s) for that date, who are from Flushing, NY and who
enrolled prior to February 7, 2019.
(2 Rows)  */

SELECT
    s.First_Name,
    s.Last_Name,
    e.Enroll_Date,
    COUNT(e.Enroll_Date) AS 'Enrollment Count'
FROM
```

```
    Student s
        JOIN
    Enrollment e ON s.Student_Id = e.Student_Id
        JOIN
    ZipCode zc ON zc.Zip = s.Zip
WHERE
    zc.City = 'Flushing' AND zc.State = 'NY'
        AND e.Enroll_Date < '2019-02-07'
GROUP BY s.First_Name , s.Last_Name , e.Enroll_Date
ORDER BY s.First_Name , s.Last_Name;

/*  19. Provide a listing of course numbers, descriptions, and formatted (with $)
costs, that
include projects (Grade_type_code = 'PJ') as a part of their grading criteria.
(Include ungraded projects too, this means you will need to use grade_type_weight
instead of grade.)
(9 rows)  */

SELECT
    c.Course_No, Description, CONCAT('$', Cost) AS 'Cost'
FROM
    Course c
        JOIN
    Section s ON c.Course_No = s.Course_No
        JOIN
    Grade_Type_Weight gtp ON s.Section_ID = gtp.Section_ID
WHERE
    gtp.Grade_Type_Code = 'PJ'
        OR gtp.Grade_Type_Code IS NULL;

/*  20. List the highest grade on the final exam (Grade_type_code = 'FI') that was
given to a
student in course 145.
(1 Row)  */

SELECT
    MAX(Numeric_Grade) AS 'Highest Grade on the Final Exam'
FROM
    Grade g
        JOIN
    Section s ON s.Section_id = g.Section_id
WHERE
    g.Grade_Type_Code = 'FI'
        AND s.Course_No = 145;
```