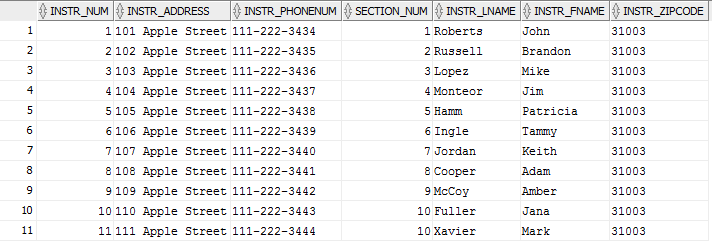
# Queries

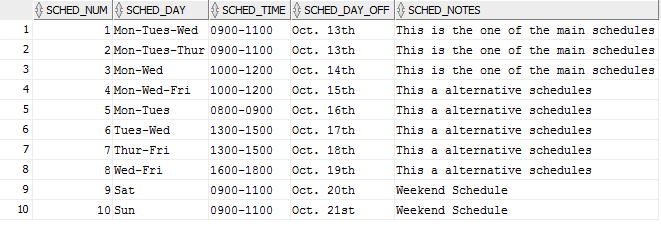
/\*1. Select all columns and all rows from one table.\*/

SELECT /\*fixed\*/ \* FROM Instr\_Info;



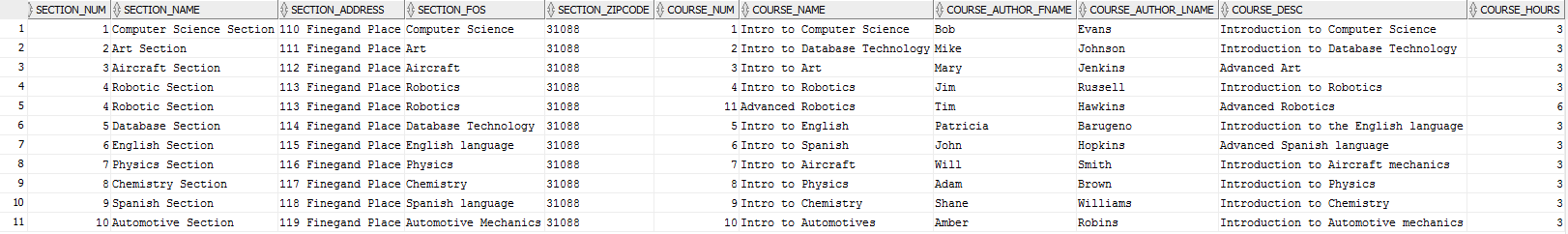
/\*2. Select 5 columns and all rows from one table. \*/

SELECT /\*fixed\*/ Sched\_Num, Sched\_Day, Sched\_Time, Sched\_Day\_Off, Sched\_Notes FROM Class\_Sched;



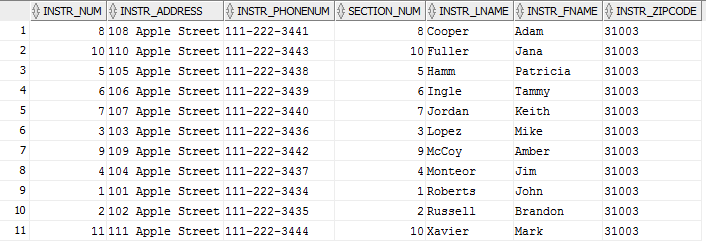
/\*3. Select all columns and all rows from 2 tables (need a join).\*/

SELECT /\*fixed\*/ \* FROM Section\_Info INNER JOIN Course\_List USING (Section\_Num);



/\*4. Select and Order data retrieved from one table \*/

SELECT /\*fixed\*/ \* FROM Instr\_Info ORDER BY Instr\_LName;



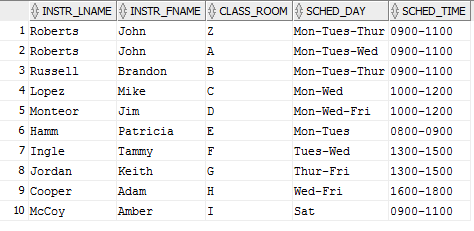
/\*5. Select 5 columns and 10 rows from 3 tables (need joins).\*/

SELECT /\*fixed\*/ a.Instr\_LName, a.Instr\_FName, b.Class\_Room, c.Sched\_Day, c.Sched\_Time

FROM (Instr\_Info a INNER JOIN Instr\_Classes b ON a.Instr\_Num=b.Instr\_Num)

INNER JOIN Class\_Sched c ON b.Sched\_Num=c.Sched\_Num

WHERE ROWNUM < 11;

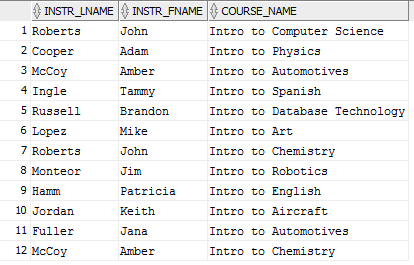


/\*6. Select distinct rows using joins using 3 tables.\*/

SELECT DISTINCT /\*fixed\*/ a.Instr\_LName, a.Instr\_FName, c.Course\_Name

FROM (Instr\_Info a INNER JOIN Instr\_Classes b ON a.Instr\_Num=b.Instr\_Num)

INNER JOIN Course\_List c ON b.Course\_Num=c.Course\_Num;

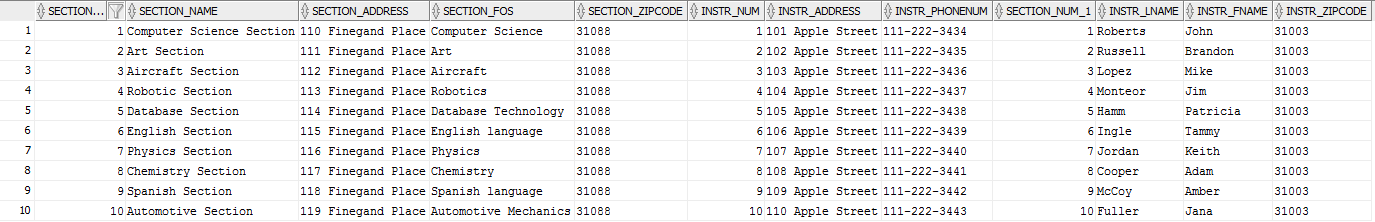


/\*7. Select all columns and 10 rows from 2 tables (need a join).\*/

SELECT /\*fixed\*/ \* FROM Section\_Info INNER JOIN Instr\_Info ON

Section\_Info.Section\_Num=Instr\_Info.Section\_Num

WHERE ROWNUM < 11;

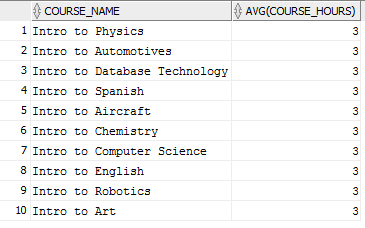


/\*8. Use group by & having in a select statement using one or more table(s).\*/

SELECT /\*fixed\*/ Course\_Name, AVG(Course\_Hours) FROM Course\_List

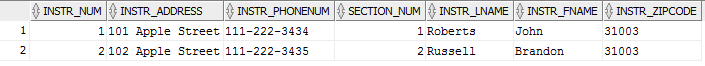
GROUP BY Course\_Name

HAVING AVG(Course\_Hours) = 3;



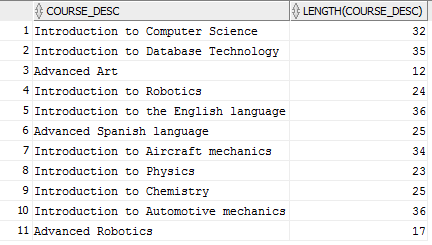
/\*9. Use a IN clause to select data from one or more tables\*/

SELECT /\*fixed\*/ \* FROM Instr\_Info WHERE Instr\_LName IN ('Roberts', 'Russell');



/\*10. Select Length of one column from one table (use Length function)\*/

SELECT /\*fixed\*/ Course\_Desc, LENGTH (Course\_Desc) FROM Course\_List;



/\*11. Use a column alias\*/

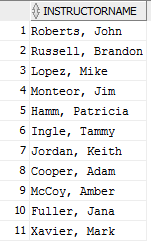
SELECT /\*fixed\*/ Instr\_FName AS InstructorName FROM Instr\_Info;



/\*12. Perform an advanced query of your choice from chapter 8 (Database Systems Text Book - Coronel, Morris & Rob)\*/

/\*Return Instructor Full Name in following format: Last, First\*/

SELECT /\*fixed\*/ Instr\_LName || ', ' || Instr\_FName AS InstructorName FROM Instr\_Info;



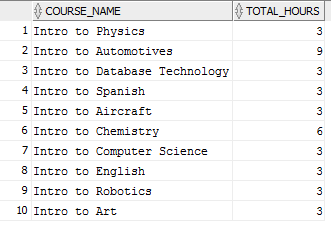
/\*13. Use an aggregate function and perform another query\*/

/\*Return total number of course hours taught for each active class\*/

SELECT /\*fixed\*/ Course\_Name, SUM(Course\_Hours) AS Total\_Hours

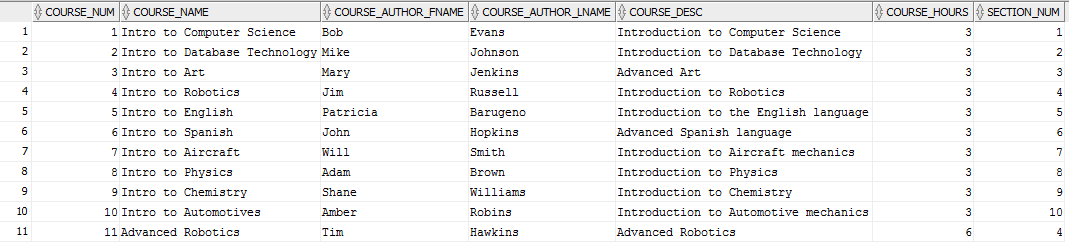
FROM Course\_List INNER JOIN Instr\_Classes ON Course\_List.Course\_Num=Instr\_Classes.Course\_Num

GROUP BY Course\_Name;



/\*14. Use the UPDATE command and change some data.\*/

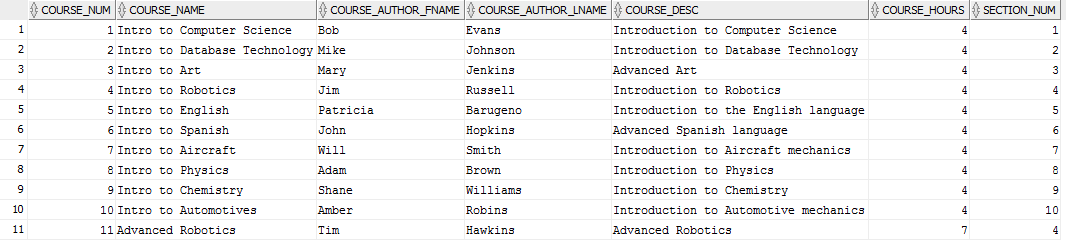
SELECT /\*fixed\*/ \* FROM Course\_List;



UPDATE Course\_List SET Course\_Hours=Course\_Hours+1;

11 rows updated.

SELECT /\*fixed\*/ \* FROM Course\_List;



ROLLBACK;

Rollback complete.

/\*15. Write an advanced SQL statement with a type I subquery (chapter 8).\*/

/\*Return list of Instructors teaching classes with "Auto" in course name using Type I query\*/

SELECT /\*fixed\*/ Instr\_Num, Instr\_LName, Instr\_FName

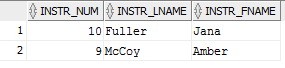
FROM Instr\_Info

WHERE Instr\_Num IN

(SELECT Instr\_Num FROM Instr\_Classes INNER JOIN Course\_List

ON Instr\_Classes.Course\_Num=Course\_List.Course\_Num

WHERE Course\_Name LIKE '%Auto%');



/\*16. Write an advanced SQL statement with type II subquery (chapter 8).\*/

/\*Return list of Instructors teaching classes with "Auto" in course name using Type II query\*/

SELECT /\*fixed\*/ Instr\_Num, Instr\_LName, Instr\_FName

FROM Instr\_Info

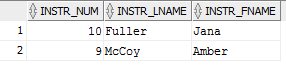
WHERE EXISTS

(SELECT \* FROM Instr\_Classes INNER JOIN Course\_List

ON Instr\_Classes.Course\_Num=Course\_List.Course\_Num

WHERE Instr\_Info.Instr\_Num=Instr\_Classes.Instr\_Num

AND Course\_Name LIKE '%Auto%');



/\*17. Perform additional advanced SQL statement. (chapter 8).\*/

/\*Return a list of courses and how many times they are currently being taught\*/

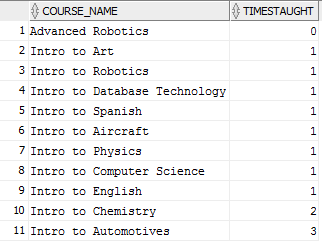
SELECT /\*fixed\*/ Course\_Name, COUNT(Instr\_Classes.Course\_Num) AS TimesTaught

FROM Course\_List LEFT JOIN Instr\_Classes

ON Instr\_Classes.Course\_Num=Course\_List.Course\_Num

GROUP BY Course\_Name

ORDER BY TimesTaught;



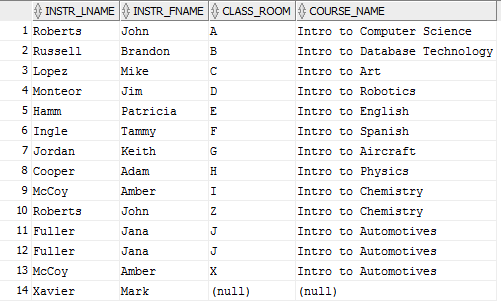
/\*18. Perform additional advanced SQL statement. (chapter 8).\*/

/\*Return all Instructor names, class rooms, and courses taught\*/

SELECT /\*fixed\*/ a.Instr\_LName, a.Instr\_FName, b.Class\_Room, c.Course\_Name

FROM (Instr\_Info a LEFT JOIN Instr\_Classes b ON a.Instr\_Num=b.Instr\_Num)

LEFT JOIN Course\_List c ON b.Course\_Num=c.Course\_Num;



/\*19. Perform additional advanced SQL statement. (chapter 8).\*/

/\*Return a list of sections not currently offering any courses\*/

SELECT /\*fixed\*/ Section\_Name

FROM Section\_Info

WHERE NOT EXISTS

(SELECT Section\_Num FROM Course\_List

WHERE Course\_List.Section\_Num=Section\_Info.Section\_Num);



/\*20. Perform additional advanced SQL statement. (chapter 8).\*/

/\*5 table join-Show all courses being taught with Section & Instructor Name,

Class Room, and Schedule; ordered by Section and Instructor last name\*/

SELECT /\*fixed\*/ e.Section\_Name, a.Instr\_LName || ', ' || a.Instr\_FName AS Instructor,

b.Class\_Room, c.Course\_Name, d.Sched\_Day || ', ' || d.Sched\_Time AS Schedule

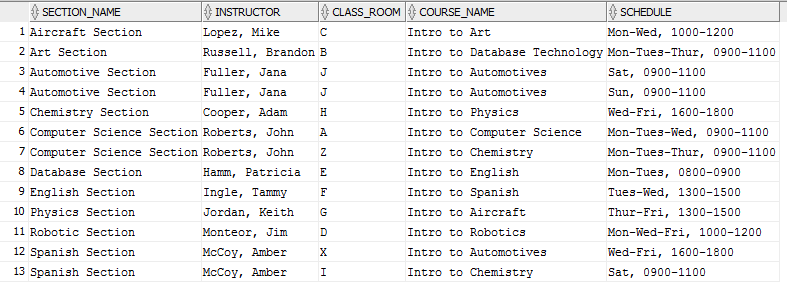
FROM (((Instr\_Info a INNER JOIN Instr\_Classes b ON a.Instr\_Num=b.Instr\_Num)

INNER JOIN Course\_List c ON b.Course\_Num=c.Course\_Num)

INNER JOIN Class\_Sched d ON b.Sched\_Num=d.Sched\_Num)

INNER JOIN Section\_Info e ON a.Section\_Num=e.Section\_Num

ORDER BY e.Section\_Name, a.Instr\_LName;



# Conclusion

This was a challenging project and I learned quite a lot from it. The more important things I felt I learned are as follows:

1. SQL Developer is a very powerful tool for database development. When I first started using it I thought it was just a simple tool for creating SQL statements. However, I quickly realized it is a very effective management tool and encompasses everything needed to create and maintain databases. Being able to utilize the query builder when creating select statements made creating queries much more efficient. Additionally, having a bird’s eye view of all the tables proved an effective multiplier in development.
2. I was unfamiliar with triggers before I began working on this project. I now know triggers are very powerful miniature programs embedded within a SQL file which can fire on specific events, similar to event-driven programming. In my project I was able to use a trigger to print a message to the screen when a new row of data was inserted into a table.
3. I knew of the JOIN operator before the beginning of this project, but this project has deepened my understanding of their true power. For example, I was able to create a join between two tables and then use a subquery to return a list of Instructors teaching classes about “Auto” with the LIKE operator. I was also able to use a LEFT JOIN and aggregate function together to compile a list of how many times a course was being taught.
4. I have built many databases in the past, but before this project creating an ERD was almost an afterthought. I now understand having an ERD with a properly laid out database design is a crucial step before you begin writing code. I referenced my ERD during all parts of this project and it proved to make my coding more efficient.
5. My Instr\_Classes table is a weak entity. With that I was unsure what would happen if I deleted a referenced entity instance in my parent tables. During this project I learned about the “ON DELETE CASCADE” operator. This is a very important operator as it will ensure any entity instance in the child table is deleted if their associated parent table entity instance is deleted.