G51DBI Lab Week 4: SQL II

INTRODUCTION

The lab for week 4 contains a set of exercises the purpose of which is to acquaint you with the basic SELECT, as well as with more complex SELECT statements including Subqueries. Remember that there is often more than one solution to a query problem; the most important thing is that your results are correct.

We will start with a variety of queries you can accomplish using SQL's SELECT statement. If you have completed the exercises of Week 3 then your database should contain tables for Artist and CD records. In order to add some variety for the queries, we should add a few more CDs. If you'd like more practice with INSERT and UPDATE operations then try to match your tables to these:

```
SELECT * FROM Artist;
```

artID	artName
6	Animal Collective
3	Deadmau5
7	Kings of Leon
4	Mark Ronson
5	Mark Ronson & The Business Intl
8	Maroon 5
2	Mr. Scruff
1	Muse

```
SELECT * FROM CD;
```

cdID	artID	cdTitle	cdPrice	cdGenre	cdNumTracks
1	1	Black Holes and Revelations	9.99	Rock	NULL
2	1	The Resistance	11.99	Rock	NULL
3	2	Ninja Tuna	9.99	Electronica	NULL
4	3	For Lack of a Better Name	9.99	Electro House	NULL
5	4	Version	12.99	Pop	NULL
6	5	Record Collection	11.99	Alternative Rock	NULL
7	6	Merriweather Post Pavilion	12.99	Electronica	NULL
8	7	Only By The Night	9.99	Rock	NULL
9	7	Come Around Sundown	12.99	Rock	NULL
10	8	Hands All Over	11.99	Pop	NULL

You may have different ID values, **do not worry about this**. If you feel you don't need to practice any more insertions or updates, then you can recreate the exact copy of these tables by using the file **setupWeek4.txt**. To display the content of each table use the SELECT statements:

```
SELECT * FROM Artist;
```

```
SELECT * FROM CD;
```

SQL SELECT

SQL's SELECT statement has a number of different options, some of which we've already seen. The simplest form of a SELECT statement returns all of the information from a single table. The general format is:

```
SELECT * FROM table-name;
```

There are more options to build up more complex queries. In this exercise we shall cover:

- SELECTing specific columns from a table;
- Using WHERE clauses to select specific rows;
- SELECTing from multiple tables;

Exercise: Write a SELECT statement to output all of the information from the Artist table

SELECTING SPECIFIC COLUMNS

Often tables in a database have many columns, and we only require a few. We can select useful ones and use aliases to make them more understandable. The aliases give the columns new names to make them easier to understand. Let's try this with our CD table:

```
SELECT cdTitle AS Title, cdGenre AS Genre FROM CD;
```

If you run the command above, you will see that the two columns returned are now named "Title" and "Genre", rather than "cdTitle" and "cdGenre".

Exercise: Write SQL Statements to do the following:

- List the names of the artists in the database
- List the genres of the CDs in the database and call the result "genre"
- List the titles and prices of the CDs in the database

You will find that the genre output contains multiple rows for the same genres. You can avoid this by using the `DISTINCT` keyword in your select statement:

```
SELECT DISTINCT cdGenre AS Genre FROM CD;
```

Exercise: Write a SELECT statement to find the artist IDs and genres from the CD table, without duplicate entries.

USING WHERE CLAUSES

By using a WHERE clause, you can return rows from tables that match specific criteria. Some examples of simple WHERE clauses using a single test are:

- `SELECT * FROM CD WHERE cdPrice > 10.00;`
- `SELECT * FROM CD WHERE cdGenre = 'Rock';`
- `SELECT * FROM CD WHERE cdGenre <> 'Rock';`

Exercise: Write SQL statements to do the following:

- Find a list of all information on Electronica CDs [Output 1]
- Find a list of the titles of all CDs that cost less than 10.00 [Output 2]

More complex conditions can be created using the operators AND, OR and NOT. Brackets can be used to control the order of evaluation. For example:

```
SELECT * FROM CD WHERE (cdPrice < 10.00) AND NOT (cdGenre =
'Rock');
```

Will return a list of CDs that cost less than 10.00, but are not Rock CDs.

Exercise: Write SQL queries to:

- Find a list of all information on CDs that have the genre 'Pop' or cost more than 10.00 [Output 3]
- Find a list of titles of CDs that are Rock albums costing between 10.00 and 12.00 [Output 4]
- Find a list of the titles of CDs that cost less than 10.00, and are either "Rock", or "Electronica". [Output 5]

SELECTING FROM MULTIPLE TABLES

It is often useful to combine information from several tables in a single query. Usually we will use a where clause to clarify which results we want, that is not always the case however. The following example will select every combination of rows from Artist and CD:

```
SELECT * FROM Artist, CD;
```

This is a CROSS JOIN, sometimes referred to as a product. In this case, the select query will return 10 * 8 rows, or 80 rows. This is far too many to be useful in this case, we should use a WHERE clause.

Exercise: Write an SQL query to:

- Return all the information from Artist and CD where the artID records are the same for both tables [Output 6].
- Find a list of the titles of all CDs by Muse [Output 7].

SUBQUERIES (USING EXISTS, IN, ANY/ALL OPTIONS)

Combining multiple tables can lead to extremely complex queries very quickly. It is sometimes easier to use subqueries to pass values into other queries. For example, suppose we want to find out a list of CDs that cost the same as 'Ninja Tuna':

```
SELECT cdTitle FROM CD WHERE cdPrice =
     (SELECT cdPrice FROM CD WHERE cdTitle = 'Ninja Tuna');
```

It is important to remember that there are various ways of comparing values with the results of a subquery. What you do is dependent not only on what results you want, but also how many values you can expect the subquery to return. In general:

1. If the subquery returns a single value, you can use the normal conditional operators (=, <, >, <>, etc.);
2. If the subquery returns multiple values, you can use IN, ANY, ALL and NOT to compare a single value to a set;
3. You can use (NOT) EXISTS to see if a set is empty or not

For example, to find a list of all CDs of the same genre as any that the artist "Mark Ronson%" has produced, we can use the following:

```
SELECT cdTitle FROM CD WHERE cdGenre IN
     (SELECT cdGenre FROM CD, Artist WHERE Artist.artID =
          CD.artID AND artName LIKE "Mark Ronson%");
```

Exercise: Write the following queries:

- Use a subquery to find a list of CDs that have the same genre as 'The Resistance' [Output 8].
- Use IN to find a list of the titles of albums that are the same price as any 'Pop' album [Output 9].
- Use ANY to find the titles of CDs that cost more than at least one other CD [Output 10].
- Use ALL to find a list of CD titles that cost more or the same as all other CDs [Output 11].
- Use EXISTS to find a list of Artists who produced an "Electronica" CD [Output 12].

Check carefully your last output (corresponding to Output 12). Is it actually true that the list of Artist you obtained produced and "Electronica" album? Do you need to add an extra condition?

Exercise: Write the following queries:

- Find the names of Artists who have albums cheaper than 10 pounds [Output 13].
- Find names of CDs produced by those Artists who have a single word as their name [Output 14].
- Find all information about CDs which are cheaper than others in the 'Rock' and 'Pop' categories only [Output 15].

Answer Sheet for G51DBI Lab Week 4

Name: ID:

Please provide your SQL Code for Exercises 1-15 described above.

OUTPUT for Exercise 1 **[0.5 marks]**

OUTPUT for Exercise 2 **[0.5 marks]**

OUTPUT for Exercise 3 **[0.5 marks]**

OUTPUT for Exercise 4 **[0.5 marks]**

OUTPUT for Exercise 5 **[0.5 marks]**

OUTPUT for Exercise 6 **[0.5 marks]**

OUTPUT for Exercise 7 **[0.5 marks]**

OUTPUT for Exercise 8 **[0.5 marks]**

OUTPUT for Exercise 9 **[0.5 marks]**

OUTPUT for Exercise 10 **[0.5 marks]**

OUTPUT for Exercise 11 **[0.5 marks]**

OUTPUT for Exercise 12 **[0.5 marks]**

OUTPUT for Exercise 13 **[0.5 marks]**

OUTPUT for Exercise 14 **[0.5 marks]**

OUTPUT for Exercise 15 **[0.5 marks]**