

[Return to "Data Foundations" in the classroom](#)

Music SQL Database

REVIEW

CODE REVIEW 4

ANNOTATIONS 1

HISTORY

▼ SQL_Queries.txt 4

```
1 -- Insight #1 ---  
2  
3 SELECT i.BillingCountry, SUM(il.UnitPrice*il.Quantity) Total_Spent  
4 FROM Invoice i  
5 JOIN InvoiceLine il
```

```
5
6   ON i.InvoiceId = il.InvoiceId
7 JOIN Track t
8   ON il.TrackId = t.TrackId
9 JOIN Genre g
10  ON t.GenreId = g.GenreId
11 WHERE g.Name = 'Rock'
12 GROUP BY 1
13 ORDER BY 2 DESC
14 LIMIT 10;
```

AWESOME

Solid logic here!

```
15
16
17 -- Insight #2 ---
18
19 SELECT g.Name, SUM(il.UnitPrice*il.Quantity) total_spent, i.InvoiceDate
20 FROM Invoice i
21 JOIN InvoiceLine il
22   ON i.InvoiceId = il.InvoiceId
23 JOIN Track t
24   ON il.TrackId = t.TrackId
25 JOIN Genre g
26   ON t.GenreId = g.GenreId
27 WHERE i.InvoiceDate > '2012-12-31'
28 GROUP BY 1
29 ORDER BY 2 DESC
30 LIMIT 5;
31
```

AWESOME

Another flawless one here!

```

33 -- Insight #3 ---
34
35 SELECT g.name, SUM(il.UnitPrice*il.Quantity) total_spent,
36        CASE WHEN sum(il.UnitPrice*il.Quantity) > 350 THEN 'Top Genre'
37        WHEN sum(il.UnitPrice*il.Quantity) > 150 AND sum(il.UnitPrice*il.Quantity) < 350 THEN 'Middle Genre'
38        ELSE 'Low Genre' END AS Genre_Category
39 FROM InvoiceLine il
40 JOIN Track t
41     ON il.TrackId = t.TrackId
42 JOIN Genre g
43     ON t.GenreId = g.GenreId
44 GROUP BY g.Name
45 ORDER BY 2 DESC

```

AWESOME

This is the kind of queries we were expecting to see. Fantastic use of conditionals to create segments!

```

46
47
48 -- Insight #4 ---
49
50 SELECT DISTINCT strftime('%Y', i.InvoiceDate) order_date, SUM(il.UnitPrice*il.Quantity) total_spent
51 FROM Invoice i
52 JOIN InvoiceLine il
53     ON i.InvoiceId = il.InvoiceId
54 JOIN Track t
55     ON il.TrackId = t.TrackId
56 JOIN Genre g
57     ON t.GenreId = g.GenreId
58 WHERE i.BillingCountry = 'United Kingdom' AND g.name = 'Rock'
59 GROUP BY 1
60

```

AWESOME

Great use of strftime here!

61
62
63

RETURN TO PATH

Rate this review

