Appendix I

Data Analysis Checkpoint Questions and Solutions

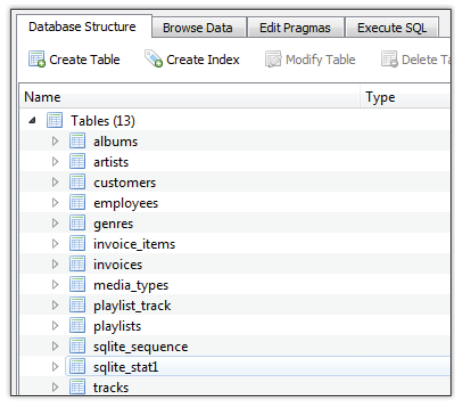
Chapter 3 Data Analysis Checkpoint

Using the Database Structure tab and the Browse Data tab, try to answer the following questions:

Question 1: How many tables are in our database?

Solution: Looking at the Database Structure tab in DB Browser, the number of tables is calculated for us and presented in parentheses `()`. There are thirteen tables in this database.

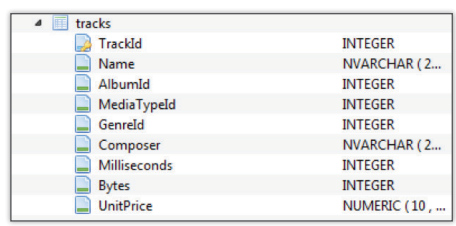
fig. 129



Question 2: How many columns does the table named *tracks* have?

Solution: For any of the tables listed, we can click on the small right-facing triangle to see the columns for that table.

fig. 130



In this example, we observe that the table called *tracks* has nine columns.

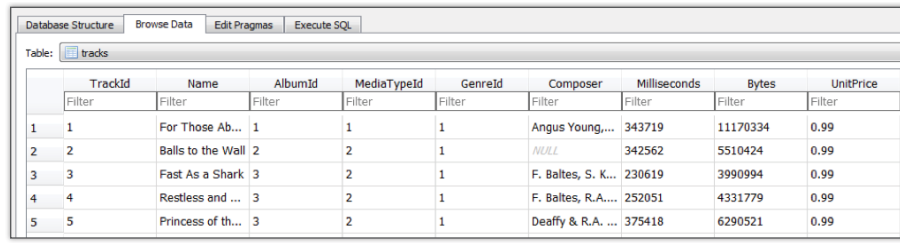
Question 3: What are some of the data types in this table?

Solution: If we look at the image from the previous question, we can see that the `TrackId` column accepts data of the type `INTEGER` and the `Name` column accepts data of the type `NVARCHAR`. The rest of the columns are also `INTEGER` and `NVARCHAR` except for `UnitPrice`, which is a `NUMERIC` data type.

Question 4: What fields are contained in the *tracks* table?

Solution: Now we can swap to the Browse Data tab and actually look at the table. We need to make sure to select the *tracks* table in the drop-down menu. Looking at the data in the table shows us why an `INTEGER` data type is used for columns like `TrackId` and `AlbumId`, while a character data type makes more sense for the `Name` and `Composer` columns. Finally, for `UnitPrice`, we needed something with decimals, so the integer data type wouldn't have been sufficient for this column.

fig. 131



	TrackId	Name	AlbumId	MediaTypeId	GenreId	Composer	Milliseconds	Bytes	UnitPrice
	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	1	For Those Ab...	1	1	1	Angus Young,...	343719	11170334	0.99
2	2	Balls to the Wall	2	2	1	*NULL*	342562	5510424	0.99
3	3	Fast As a Shark	3	2	1	F. Baltes, S. K...	230619	3990994	0.99
4	4	Restless and ...	3	2	1	F. Baltes, R.A....	252051	4331779	0.99
5	5	Princess of th...	3	2	1	Deaffy & R.A. ...	375418	6290521	0.99

Chapter 4 Data Analysis Checkpoint

Question 1: How many customers' last names begin with *B*?

Solution: In order to answer this question, we can first write a query to display the specific information we are looking for. In this case, we are interested in last names. Last names are contained in the *customers* table under the field name `LastName`.

If we simply want a query to display all the last names, we can do this:

```
SELECT
   LastName
FROM
   customers
```

That will give us a field of all the last names, but they aren't in any particular order. To alphabetize them we can use the `ORDER BY` statement. Note that we don't have to specify A-Z because ascending order is returned by default. If we were looking for names starting with Z, we might have included the `DESC` statement.

```
SELECT
   LastName
FROM
   customers
ORDER BY
LastName ASC
```

fig. 132

	LastName
1	Almeida
2	Barnett
3	Bernard
4	Brooks
5	Brown
6	Chase
7	Cunningham
8	Dubois
9	Fernandes
10	Francis
...	

Now our results are alphabetized and we can easily see that four of the entries start with *B*. Note that we are still using observation to determine how many entries start with *B*. Other ways to do this will be explored further on.

Question 2: When sorted in descending order, which company appears at the top, in the *customers* table?

Solution: This time we are looking for the `Company` field instead of the `LastName` field. As mentioned in the last question, all we have to do is change the last part of our query to specify descending order.

```
SELECT
   Company
FROM
   customers
ORDER BY
   Company DESC
```

Doing this yields the following result:

fig. 133

	Company
1	Woodstock Discos
2	Telus
3	Rogers Canada
4	Riotur
5	Microsoft Corporation
6	JetBrains s.r.o.
7	Google Inc.
8	Embraer - Empresa Brasileira de Aeronáutica S.A.
9	Banco do Brasil S.A.
10	Apple Inc.
...	

Again, we can observe that Woodstock Discos is the first company listed in descending order.

Question 3: How many customers do not have a postal code listed?

Solution: We could answer this question by scrolling through the data on Browse Data, but there is a better way. Using a `SELECT` statement, we can list all the data in ascending order as we have done previously. But this time, we want to list more than just one column, so we can see what customer names have no postal code data. So we can choose `FirstName`, `LastName`, and `PostalCode`, then order the results by `PostalCode`.

```
SELECT
   FirstName,
   LastName,
   PostalCode
FROM
    customers
ORDER BY
   PostalCode
```

This shows us four entries that do not have postal data, as designated by the null value in the `PostalCode` column (Figure 134).

fig. 134

	FirstName	LastName	PostalCode
1	João	Fernandes	NULL
2	Madalena	Sampaio	NULL
3	Hugh	O'Reilly	NULL
4	Luís	Rojas	NULL
5	Stanisław	Wójcik	00-358
6	Lucas	Mancini	00192
7	Terhi	Hämäläinen	00530
8	Eduardo	Martins	01007-010
9	Alexandre	Rocha	01310-200
10	Bjørn	Hansen	0171
...			

If we were to list these in descending order, we would have to scroll to the bottom to see the null values.

Chapter 5 Data Analysis Checkpoint

Question 1: Create a query for the *invoices* table that includes a `CASE` statement that labels all sales from billing country USA as "Domestic Sales" and all other sales as "Foreign Sales." Label your new field as `SalesType` after your `END AS` statement.

Solution: To display this information, we combine what we learned about filtering records by text with our `CASE` statement. Since we are categorizing our `CASE` statement by billing country, we will have to include that field in our `SELECT` statement.

```
SELECT
   InvoiceDate,
   BillingAddress,
   BillingCity,
   BillingCountry,
   Total,
   CASE
   WHEN BillingCountry = 'USA' THEN 'Domestic Sales'
   ELSE 'Foreign Sales'
   END AS SalesType
FROM
   invoices
```

fig. 135

	InvoiceDate	BillingAddress	BillingCity	BillingCountry	Total	PurchaseType
1	1/1/2009 0:00	Theodor-Heuss-Straße 34	Stuttgart	Germany	1.98	Foreign Sales
2	1/2/2009 0:00	Ullevålsveien 14	Oslo	Norway	3.96	Foreign Sales
3	1/3/2009 0:00	Grétrystraat 63	Brussels	Belgium	5.94	Foreign Sales
4	1/6/2009 0:00	8210 111 ST NW	Edmonton	Canada	8.91	Foreign Sales
5	1/11/2009 0:00	69 Salem Street	Boston	USA	13.86	Domestic Sales
6	1/19/2009 0:00	Berger Straße 10	Frankfurt	Germany	0.99	Foreign Sales
7	2/1/2009 0:00	Barbarossastraße 19	Berlin	Germany	1.98	Foreign Sales
8	2/1/2009 0:00	8, Rue Hanovre	Paris	France	1.98	Foreign Sales
9	2/2/2009 0:00	9, Place Louis Barthou	Bordeaux	France	3.96	Foreign Sales
10	2/3/2009 0:00	3 Chatham Street	Dublin	Ireland	5.94	Foreign Sales
...						

Question 2: Order this data by the new field `SalesType`.

Solution: To show all domestic sales in one group and all foreign sales in another group, we simply add an `ORDER BY` (using our new field) to our existing query:

```
SELECT
   InvoiceDate,
   BillingAddress,
   BillingCity,
   BillingCountry,
   Total,
   CASE
   WHEN BillingCountry = 'USA' THEN 'Domestic Sales'
   ELSE 'Foreign Sales'
   END AS SalesType
FROM
   invoices
ORDER BY
   SalesType
```

Figure 136 shows the results for this query.

fig. 136

	InvoiceDate	BillingAddress	BillingCity	BillingCountry	Total	SalesType
1	1/11/2009 0:00	69 Salem Street	Boston	USA	13.86	Domestic Sales
2	2/19/2009 0:00	1600 Amphitheatre Parkway	Mountain View	USA	0.99	Domestic Sales
3	3/4/2009 0:00	1 Microsoft Way	Redmond	USA	1.98	Domestic Sales
4	3/4/2009 0:00	1 Infinite Loop	Cupertino	USA	1.98	Domestic Sales
5	3/5/2009 0:00	801 W 4th Street	Reno	USA	3.96	Domestic Sales
6	3/6/2009 0:00	319 N. Frances Street	Madison	USA	5.94	Domestic Sales
7	4/14/2009 0:00	1 Infinite Loop	Cupertino	USA	13.86	Domestic Sales
8	6/6/2009 0:00	1 Microsoft Way	Redmond	USA	3.96	Domestic Sales
9	6/7/2009 0:00	801 W 4th Street	Reno	USA	5.94	Domestic Sales
10	6/10/2009 0:00	1033 N Park Ave	Tucson	USA	8.91	Domestic Sales
...						

Question 3: How many invoices from Domestic Sales were over $15?

Solution: We can use the same query again, but this time add a `WHERE` clause and `AND` to include both the numeric and text parameters.

```
SELECT
   InvoiceDate,
   BillingAddress,
   BillingCity,
   BillingCountry,
   Total,
   CASE
   WHEN BillingCountry = 'USA' THEN 'Domestic Sales'
   ELSE 'ForeignSales'
   END AS SalesType
FROM
   invoices
Where
   SalesType = "Domestic Sales" AND Total > 15
```

fig. 137

	InvoiceDate	BillingAddress	BillingCity	BillingCountry	Total	SalesType
1	3/21/2010 0:00	162 E Superior Street	Chicago	USA	15.86	Domestic Sales
2	5/29/2011 0:00	319 N. Frances Street	Madison	USA	18.86	Domestic Sales
3	8/5/2012 0:00	2211 W Berry Street	Fort Worth	USA	23.86	Domestic Sales

Chapter 6 Data Analysis Checkpoint

Question 1: Using DB Browser and the Browse Data tab or the entity relationship diagram on page 95, view the *tracks* table. Identify which fields in that table are foreign keys in another table. Based on the foreign keys you have identified, which tables are related to the *tracks* table?

Solution: Looking at the *tracks* table, we see three fields with integer values that appear to be foreign keys.

fig. 138

	TrackId	Name	AlbumId	MediaTypeId	GenreId	Composer	Milliseconds	Bytes	UnitPrice
	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	1	For Those Ab...	1	1	1	Angus Young,...	343719	11170334	0.99
2	2	Balls to the Wall	2	2	1	*NULL*	342562	5510424	0.99
3	3	Fast As a Shark	3	2	1	F. Baltes, S. K...	230619	3990994	0.99
4	4	Restless and ...	3	2	1	F. Baltes, R.A....	252051	4331779	0.99
5	5	Princess of th...	3	2	1	Deaffy & R.A. ...	375418	6290521	0.99
6	6	Put The Finge...	1	1	1	Angus Young,...	205662	6713451	0.99
7	7	Let's Get It Up	1	1	1	Angus Young,...	233926	7636561	0.99
8	8	Inject The Ve...	1	1	1	Angus Young,...	210834	6852860	0.99
9	9	Snowballed	1	1	1	Angus Young,...	203102	6599424	0.99
10	10	Evil Walks	1	1	1	Angus Young,...	263497	8611245	0.99

The fields `AlbumId`, `MediaTypeId`, and `GenreId` correspond to the *albums*, *media_types*, and *genres* tables, respectively.

Question 2: Create an inner join between the *albums* and *tracks* tables and display the album names and track names in a single result set.

Solution:

```
SELECT
a.title,
   t.Name
FROM
   albums a
INNER JOIN
   tracks t
ON
a.AlbumId = t.TrackId
```

Question 3: Using the *genres* table identified in question 1, create a third inner join to join to this table and include the `Name` field from that table in your result set.

Solution:

```
SELECT
   a.title,
         t.Name,
         g.Name
FROM
   albums a
INNER JOIN
   tracks t
ON
   a.AlbumId = t.TrackId
INNER JOIN
   genres g
ON
   g.GenreId = t.GenreId
```

Chapter 7 Data Analysis Checkpoint

Question 1: Create a single-line mailing list for all US customers, including capitalized full names and full addresses with five-digit zip codes, in the following format:

FRANK HARRIS 1600 Amphitheatre Parkway, Mountain View, CA 94043

Solution: The format above is calling for the first and last names to be in all caps, so we will need the `UPPER()` function for those two fields. We use the double pipes to concatenate the rest of the fields, adding spaces and commas where needed.

```
SELECT
   UPPER(FirstName) || ' ' || UPPER(LastName) || ' '
   || Address || ', ' || City || ', ' || State || ' '
   || SUBSTR(PostalCode,1,5) AS [MailingAddress]
FROM
   customers
WHERE
   Country = 'USA'
```

fig. 139

	MailingAddress
1	FRANK HARRIS 1600 Amphitheatre Parkway, Mountain View, CA 94043
2	JACK SMITH 1 Microsoft Way, Redmond, WA 98052
3	MICHELLE BROOKS 627 Broadway, New York, NY 10012
4	TIM GOYER 1 Infinite Loop, Cupertino, CA 95014
5	DAN MILLER 541 Del Medio Avenue, Mountain View, CA 94040
1	KATHY CHASE 801 W 4th Street, Reno, NV 89503
2	HEATHER LEACOCK 120 S Orange Ave, Orlando, FL 32801
3	JOHN GORDON 69 Salem Street, Boston, MA 2113
4	FRANK RALSTON 162 E Superior Street, Chicago, IL 60611
5	VICTOR STEVENS 319 N. Frances Street, Madison, WI 53703
...	

Question 2: What are the average annual sales generated by customers from the USA from all years of data available?

Solution: If we are just looking for an aggregate function for one country, we can simply select billing country and the average of the total using the `WHERE` clause to limit our results to the USA.

```
SELECT
   BillingCountry,
   AVG(Total)
FROM
   invoices
WHERE
   BillingCountry = 'USA'
```

fig. 140

	BillingCountry	AVG(Total)
1	USA	5.7479121

We can use the `ROUND()` function outside of the `AVG()` function to reduce the number of decimal places returned.

Question 3: What are the company's all-time total sales?

Solution: Since this question is asking us for the sum total of invoices, our `SELECT` statement is fairly simple.

```
SELECT
   SUM(Total)
FROM
   invoices
```

fig. 141

	SUM(Total)
1	2328.6

Question 4: Who are the top ten best customers from a revenue standpoint? *Hint:* you will need to use a join (chapter 6) to answer this question.

Solution: We have already found the total revenue. Now we are looking for the top ten customers responsible for the highest revenue. Since we are looking for data from one table that corresponds to data from another table in a one-to-one relationship, we use an inner join.

```
SELECT
   SUM(Total)AS [Revenue Total],
   c.FirstName,
   c.LastName
FROM
   invoices i
INNER JOIN
   customers c
ON
   i.CustomerId = c.CustomerId
GROUP BY c.CustomerId
ORDER BY SUM(Total) DESC
```

Chapter 8 Data Analysis Checkpoint

Question 1: How many invoices exceed the average invoice amount generated in 2010?

Solution: To answer this question we need to accomplish two tasks. First we need to find the average invoice amount generated in 2010. Secondly, we need to compare that value with every invoice in our table to see how many exceeded the average 2010 invoice value.

First let's write our subquery:

```
select
   avg(total)
from
   invoices
where
   InvoiceDate between '2010-01-01' and '2010-12-31'
```

Running this query gives us an average of $5.80; now we need to write the outer query to select invoices that are greater than the 2010 average.

```
SELECT
   InvoiceDate,
   Total
FROM
   invoices
WHERE
Total >

(select
   avg(total)
from
   invoices
where
   InvoiceDate between '2010-01-01' and '2010-12-31')
ORDER BY
   Total DESC
```

fig. 142

	InvoiceDate	Total
1	11/13/2013 0:00	25.86
2	8/5/2012 0:00	23.86
3	2/18/2010 0:00	21.86
4	4/28/2011 0:00	21.86
5	1/18/2010 0:00	18.86
6	5/29/2011 0:00	18.86
7	1/13/2010 0:00	17.91
8	9/5/2012 0:00	16.86
9	10/6/2012 0:00	16.86
10	3/21/2010 0:00	15.86
...		

Our Results Pane tells us that 179 results were returned.

If we only wanted the actual number of invoices returned, we could modify our `Total` field in our outer query to say `COUNT(Total)`.

Question 2: Who are the customers responsible for these invoices?

Solution: This problem requires joins again, to connect customer data from the *customers* table to the *invoices* table. The question itself implies a one-to-one relationship between the *customers* table and the *invoices* table. We have already selected the invoices we are interested in, so now we need to find the customers attached to those invoices. This is exactly what an inner join does. This solution is very similar to the solution to Question 1. All we have added is the inner join section so we have access to customer names as well.

```
SELECT
   i.InvoiceDate,
   i.Total,
   c.FirstName,
   c.LastName
FROM
   invoices i
INNER JOIN
   customers c
ON
   i.CustomerId = c.CustomerId
WHERE
Total >

(select
   avg(total)
from
   invoices
where
   InvoiceDate between '2010-01-01' and '2010-12-31')
ORDER BY
   Total DESC
```

Question 3: How many of these customers are from the USA?

Solution: We can modify the solution to Question 2 above to include an `AND` statement at the end of the `WHERE` clause of the outer query.

```
SELECT
   InvoiceDate,
   Total,
   BillingCountry
FROM
   invoices
WHERE
Total >

(select
   avg(total)
from
   invoices
where
   InvoiceDate between '2010-01-01' and '2010-12-31')
AND BillingCountry = 'USA'
ORDER BY
   Total DESC
```

fig. 143

	InvoiceDate	Total	BillingCountry
1	11/13/2013 0:00	25.86	USA
2	8/5/2012 0:00	23.86	USA
3	2/18/2010 0:00	21.86	USA
4	4/28/2011 0:00	21.86	USA
5	1/18/2010 0:00	18.86	USA
6	5/29/2011 0:00	18.86	USA
7	1/13/2010 0:00	17.91	USA
8	9/5/2012 0:00	16.86	USA
9	10/6/2012 0:00	16.86	USA
10	3/21/2010 0:00	15.86	USA
...			

Our Results Pane shows us that the query returned forty records.

We could use a `SUM()` function around the total if we wanted this query to return the exact number of results.

Chapter 9 Data Analysis Checkpoint

In this checkpoint we asked you to turn the following query, which compares average invoice per city against the global average, into a series of views:

```
SELECT
   BillingCity,
   AVG(Total) AS [City Average],
   (select
         avg(total)
   from
         invoices) AS [Global Average]
FROM
   invoices
GROUP BY
   BillingCity
ORDER BY
   BillingCity
```

Question 1: Take the inner query (by itself) from this `SELECT` statement and create a view from it. Save the view as `V_GlobalAverage`.

If you have been following along with the in-chapter examples, you might have already saved an average function as a view. For this exercise, make sure this new view has a new name.

Solution: We take the inner query by itself and add the view syntax on the first line.

```
CREATE VIEW V_GlobalAverage AS
select
   avg(total)
from
   invoices AS [Global Average]
```

Question 2: Remove the subquery from the code above entirely and substitute it for your newly created view `V_GlobalAverage`.

Solution: When we use a view in the `SELECT` clause, we use the asterisk symbol.

```
SELECT
   BillingCity,
   AVG(Total) AS [City Average],
(select
   *
from
V_GlobalAverage) AS [Global Average]
FROM
   invoices
GROUP BY
   BillingCity
ORDER BY
   BillingCity
```

Question 3: Save this new query as a view called `V_CityAvgVsGlobalAvg`.

Solution: We copy our code from question 2 and add the `CREATE VIEW` statement at the very top.

```
CREATE VIEW V_CityAvgVsGlobalAvg AS
SELECT
   BillingCity,
   AVG(Total) AS [City Average],
(select
   *
from
V_GlobalAverage) AS [Global Average]
FROM
   invoices
GROUP BY
   BillingCity
ORDER BY
   BillingCity
```

Question 4: Delete the view `V_GlobalAverage`. What happens to `V_CityAvgVsGlobalAvg`?

Solution: We use `DROP VIEW` to delete our view. Alternatively, we can right-click on the view from our Database Structure tab in DB Browser and delete the view that way.

```
DROP VIEW V_GlobalAverage
```

Now to see how this impacts our previous statements, we need to write a `SELECT` statement to select our virtual table.

```
V_CityAvgVsGlobalAvg
SELECT
   *
FROM
   V_CityAvgVsGlobalAvg
```

You should get the following error message:

```
no such table: main.V_GlobalAverage:
```

Chapter 10 Data Analysis Checkpoint

Question 1: Add a new customer to the database.

Solution: We first need to add our new customer to the *customers* table. A customer can exist alone without being referenced on any other table (if they didn't make a purchase yet). To start, insert a record into the *customers* table.

```
INSERT INTO
customers
VALUES ('60', 'New', 'Customer', '', '123 Day Street',
'New York', 'NY', 'USA', '11201', '(347) 525-8688', '',
'nc@gmail.com', '1');
```

We left some of the fields as null by including two single quotes next to each other. We can check our work by running a `SELECT` statement that looks for the name of the customer we just added.

```
SELECT
*
FROM
customers
WHERE
FirstName = 'New'
```

If you used a different name for your new customer, modify that value in the query accordingly.

fig. 144

	CustomerId	FirstName	LastName	Company	Address	City	State	Country	PostalCode	Phone	Fax	Email	SupportId
1	60	New	Customer		123 Day Street	New York	NY	USA	11201	(347)525-8688		nc@gmail.com	1

Question 2: Create an invoice record for this customer.

Solution: In order to create an invoice entry for our new customer, we must pay special attention to the fields in the *invoices* table that correspond to our *customers* table. For example, our invoices use the same address that appears in the *customers* table.

```
INSERT INTO
invoices
VALUES ('413', '60', '2019-10-04 00:00:00', '123 Day
Street', 'New York', 'NY', 'USA', '10201', '50.00')
```

Question 3: Remove this customer from the database.

Solution: As we mentioned in chapter 10, it is a best practice to view the data we are going to delete so that we can see what we will be deleting. In this case, the data we are deleting stretches across two tables, so we write an `INNER JOIN` statement to view all the data we have included.

```
SELECT
   c.FirstName,
   c.LastName,
   i.Total,
   i.InvoiceId
FROM
   invoices i
```

```
INNER JOIN
   customers c
ON       i.CustomerId = c.CustomerId
WHERE c.CustomerId = 60
```

Now that we have confirmed the data, we can compose the `DELETE` statement.

```
DELETE FROM
   invoices
WHERE CustomerId = 60

DELETE FROM
   customers
WHERE CustomerId = 60
```