PIM Training Program

SQL

Data Warehouse & SQL Basics

Objectives of Training

- At end of the module , the learner will be familiar with data net and the basic structure of SQL query

Agenda

- SQL
  - Intro to RDBMS
  - Intro to SQL
  - SELECT & FROM Clauses
  - Table Aliases
  - Column Aliases
  - Types of Elements
  - Concatenating
  - ORDER BY Clause
- ETL
  - Getting Started with ETL
  - Anatomy of ETL Manager
  - Creating & Running a Data Feed
  - Job Run Details Page
  - Viewing Your Jobs
  - Dependency Syntax
  - Hoot

- Intro to RDBMS

RDBMS Fundamentals

- Relational Database Management System.
- RDBMS is the basis for SQL, and for all modern database systems such as MS SQL Server, IBM DB2, Oracle, MySQL, and Microsoft Access.
- The data in RDBMS is stored in database objects called tables.
- A table is a collection of related data entries and it consists of columns and rows.

Entity Relationship Diagram

RDBMS is a collection of related tables

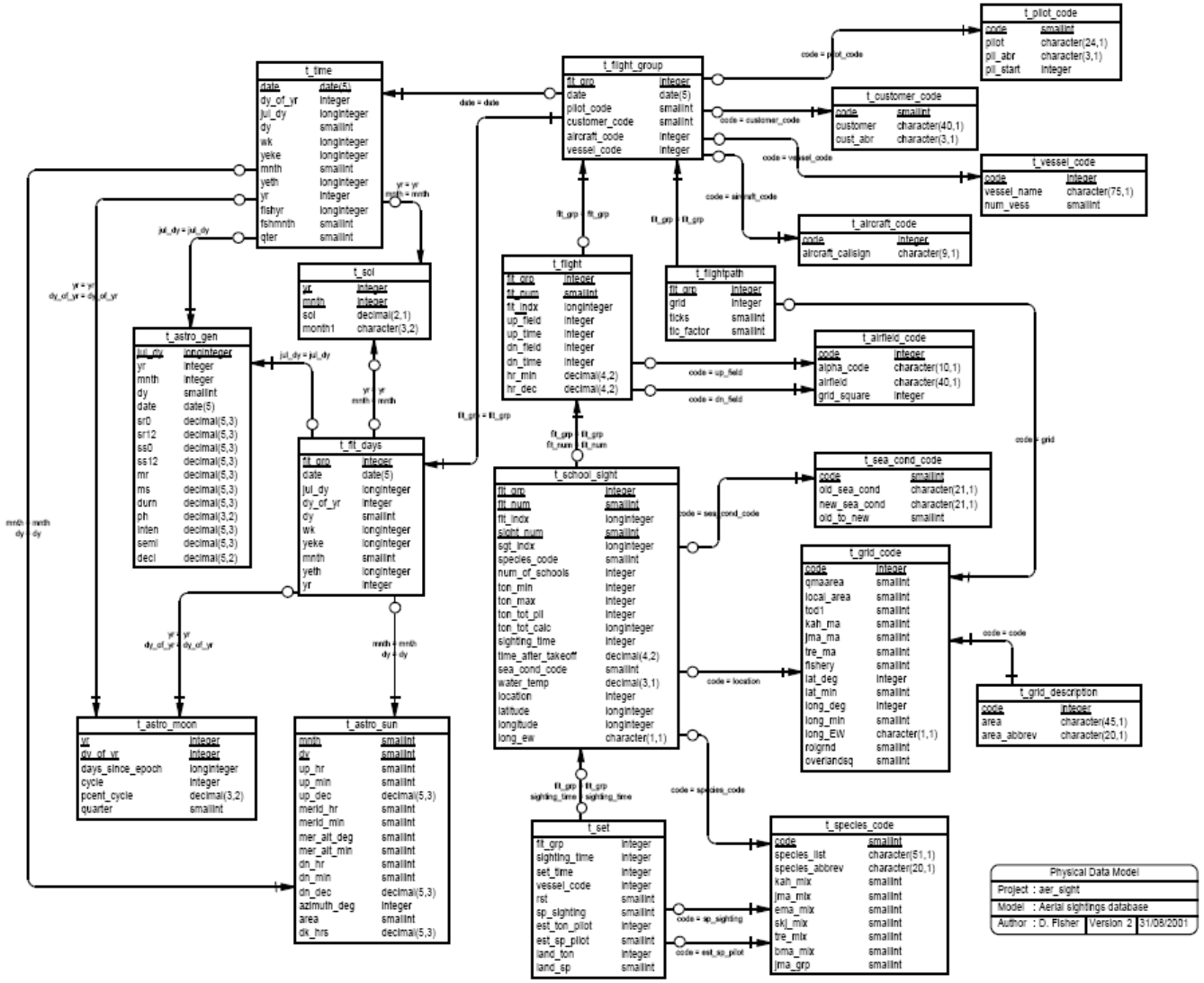

Figure 1: Entity Relationship Diagram (ERD) for the aer_sight database

Table Relationships

Primary Key
uniquely identifies each record in a database table

Foreign Key
points to a PRIMARY KEY in another table

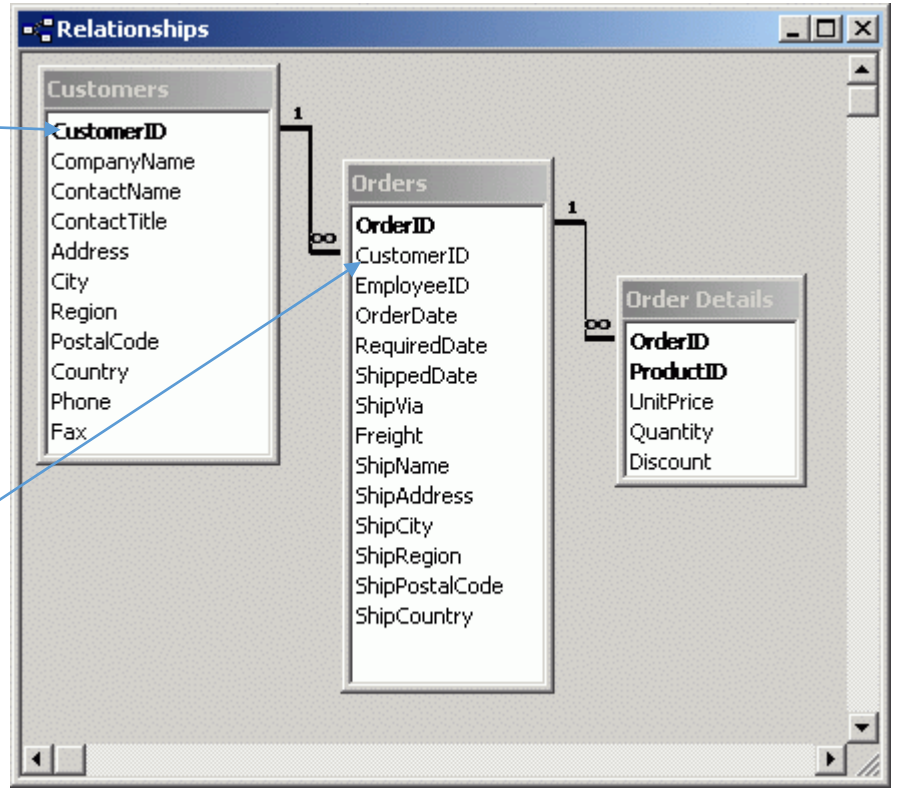

Table Structure

Column Header / Attribute Name

Field - contains column-specific data for each row

Columns

Rows

item_authority_id	cost	Original Vendor Code	Product Group Code	status	submitted_cost	Order Id	price
B000CO86EG	8.91	TOSQG	263	BO	8.91	3XLE8IHR	17.99
B000087K65	2.07	MAU6O	229		2.07	2ONSS5BH	4.75
B001KPU7TI	4.67	UNJF1	328		4.67	29I1VQWZ	93.4
1888766077	15.39	BTMB	14		15.39	4UIJ8NEB	27
B00IO2YFHK	22.03	LAGAU	328		22.03	8XYV3REI	58.8
B000C2Q1ZE	40.44	PAW95	263		40.44	3GYUCQQN	77.56
B002Z2QDNE	11.38	DHCX9	147		11.91	3OI1O2HU	59.99
B004MLWKMY	25	IROF9	201		25	49VB1SZL	49.99
B003TN6FPY	13.56	PEJZ9	325		13.56	4SQOWT8U	22.8
B001ET721G	9.42	REVL7	194		9.42	5WTPZG8N	14.98
B007Q2KZU0	10	BIGYD	21	BO	10	37D4DY4K	19.99
B005AK8WFU	8.55	AMIA9	325		10.95	7S9I4WUH	16.85
B00004Z2HD	1.11	JEBO9	469		1.11	7X2PXAOZ	2.69
B00023D368	48.71	RSYS9	199		48.71	2NJGC6WA	99.99
B002ZU4T96	5.22	MAYDV	194		5.26	7CKJDUOP	8.95
B00FOHVV98	6.5	CARMF	201		6.5	8WSCYQPJ	14.99
B00K0E92E6	21.72	ECDIF	199		21.72	878BRO4D	32.29

Amazon’s Data Warehouse

- Supports hundreds of different functional areas, from customer orders to purchase orders to email campaigns to glance views to andon cords, etc.
- Data is stored in several thousands of tables in this warehouse

Amazon's Data Warehouse contd..

- A collection of more than one multiple Tables known as a Relational Database
- Stores copies of data from Production DBs
- Each Table contains Columns of Data
- There may be multiple Tables for each Subject Area, though some Production Data may not be copied to DW
- The data is loaded and/or updated nightly
- SQL allows us to Extract data out of these tables

- Intro to SQL

Intro to SQL

- A programming language designed for managing relational databases
- SQL Queries are made up of Clauses containing Expressions and Conditions

SQL

- Can be used to:
  - Create and Manage Tables
  - Load Data Introduction Tables
  - Pull Data out of Tables

Two Required SQL Clauses

SELECT – Defines what columns to extract from the tables and include in the result set

FROM – Defines in which table(s) the data you need is found (and how they are related)

Two Required SQL Clauses

- Although SQL is very forgiving in terms of upper/lower case and extra spaces, however, it is sensitive towards tabs, line breaks and the order of the clauses.
- SELECT always comes before FROM.
- Ending your SQL statements with a semi-colon (;) is optional (but best practice).

Our First Query

```
SELECT

FROM
;
```

We'll start with the basic outline – SELECT, FROM and ;

Specify Table in FROM Clause

```
SELECT

FROM PRODUCT_GROUPS
;
```

Then we'll add the table that has the data we need to the FROM clause. In this case, the table PRODUCT_GROUPS.

Specify Column(s) in SELECT Clause

```
SELECT
*
FROM PRODUCT_GROUPS
;
```

Then we'll list the columns we want in our SELECT clause

Asterisk

```
SELECT
*
FROM PRODUCT_GROUPS
;
```

In this case, Asterisk is used to select all columns from the table.

Our First Query's Results!

This query would return a result set like this

```
SELECT
*
FROM PRODUCT_GROUPS
;
```

PRODUCT_GROUPz	DESCRIPTION	CREATION_DATE	LAST_UPDATED	DW_CREATION_DATE	DW_LAST_UPDATED	SHORT_DESC
420	Financial_Products	10-Jun-11	10-Jun-11	10-Jun-11	10-Jun-11	
424	Digital_Text_2	15-Jul-11	16-Jul-11	16-Jul-11	16-Jul-11	
425	Digital_Accessories_2	15-Jul-11	16-Jul-11	16-Jul-11	16-Jul-11	
437	Amazon_Points	4-Feb-12	4-Feb-12	4-Feb-12	4-Feb-12	
426	Publisher_Services	31-Jul-11	31-Jul-11	31-Jul-11	31-Jul-11	
251	Gourmet	4-Aug-03	4-Aug-03	4-Aug-03	4-Aug-03	Gourmt
241	Watches	4-Jun-03	4-Jun-03	4-Jun-03	4-Jun-03	Watches
236	Misc SDP	20-Dec-02	12-Dec-02	20-Dec-02	20-Dec-02	
234	Travel Store	9-Sep-02	9-Sep-02	9-Sep-02	9-Sep-02	
259	Sports Memorabilia	31-Oct-03	31-Oct-03	31-Oct-03	31-Oct-03	
258	Posters	31-Oct-03	31-Oct-03	31-Oct-03	31-Oct-03	
309	Shoes	18-Aug-05	18-Aug-05	18-Aug-05	18-Aug-05	Shoes
23	Electronics			14-Dec-99	14-Dec-99	CE
21	Toys			14-Dec-99	14-Dec-99	Toys

Specifying Columns

```
SELECT
PRODUCT_GROUP
, DESCRIPTION
FROM PRODUCT_GROUPS
;
```

We can get more specific and tell the query exactly which columns from the table PRODUCT_GROUPS we want by listing them in the SELECT clause, separated by commas

Results!

```
SELECT
PRODUCT_GROUP
, DESCRIPTION
FROM PRODUCT_GROUPS
;
```

PRODUCT_GROUP	DESCRIPTION
412	Cloud_Software_Applications
437	Amazon_Points
414	A_Drive
416	Deal_Sourcer
417	Amazon_Sourced
420	Financial_Products
424	Digital_Text_2
425	Digital_Accessories_2
251	Gourmet
241	Watches
266	Antiques
267	Musical Instruments
311	Free_Gift_Card
18	Unknown
65	Software

Now the query returns just the two columns we specified, and in the order we listed them in our SELECT clause

Comma Placement

```
SELECT
PRODUCT_GROUP
, DESCRIPTION
FROM PRODUCT_GROUPS
;
```

Notice the Comma between the two columns comes at the start of the second line, before the second column name, and not after the first column name.

Comma Placement

```
SELECT
PRODUCT_GROUP
, DESCRIPTION
FROM PRODUCT_GROUPS
;
```

This is a style choice, not a syntax requirement… but there's good reason for it.

Intro to SQL

Why?

```
SELECT
PRODUCT_GROUP
, DESCRIPTION
FROM PRODUCT_GROUPS
;
```

If we decide to remove the DESCRIPTION column in the results, the whole row from SQL can simply be deleted and avoid having to keep track of a lingering comma

Common Comma Error

```
SELECT
PRODUCT_GROUP ,
FROM PRODUCT_GROUPS
;
```

Lingering comma!

Extra commas is a common cause of errors for beginning (and sometimes advanced) SQL programmer. This little style trick helps prevent the problem. This is a strongly recommended practice.

- Table Aliases

Table Aliases

```
SELECT
PRODUCT_GROUP
, DESCRIPTION
FROM PRODUCT_GROUPS
;
```

Another best practice that isn’t necessary for single-table queries like this, but is required once we start joining 2 or more tables, is to use Table Aliases

Table Aliases

Table Aliases – A Nickname

```
SELECT
PRODUCT_GROUP
, DESCRIPTION
FROM PRODUCT_GROUPS
;
```

You start by picking a short (maybe 1-4 characters) nickname for each table in your query. Often, the first initials of each word in the table name are chosen.

Table Aliases

Table Aliases – Right after Table

```
SELECT
PRODUCT_GROUP
, DESCRIPTION
FROM PRODUCT_GROUPS pg
;
```

In this case, ‘pg’ would be a logical alias for the table PRODUCT_GROUPS

Table Aliases

Table Aliases – Before Column Names

```
SELECT
pg.PRODUCT_GROUP
, pg.DESCRIPTION
FROM PRODUCT_GROUPS pg
;
```

Add the nickname followed by a period, right in front of all the coloumn names in your query that from the mentioned table.Then you put the same nickname, followed by a period, right in front of all the column names in your query that come from that table.

Table Aliases

Table Aliases – Before Column Names

```
SELECT
pg.PRODUCT_GROUP
, pg.DESCRIPTION
FROM PRODUCT_GROUPS pg
;
```

This makes it clear from which table each column is derived.

When you start writing queries with multiple tables, and those tables might have a column in common, it becomes vital to use table aliases – so getting in the habit now will make it easier later.

- Column Aliases

Column Aliases

Column Aliases

```
SELECT
pg.PRODUCT_GROUP
, pg.DESCRIPTION
FROM PRODUCT_GROUPS pg
;
```

Another type of alias is a Column alias. These are used to alter what the column header says in your results.

Column Aliases

Column Aliases

```
SELECT
pg.PRODUCT_GROUP
, pg.DESCRIPTION
FROM PRODUCT_GROUPS pg
;
```

For example, you might want your results to call the PRODUCT_GROUP column "GL_CODE" and the DESCRIPTION column "GL_NAME".

Column Aliases

Column Aliases

```
SELECT
pg.PRODUCT_GROUP as GL_CODE
, pg.DESCRIPTION
FROM PRODUCT_GROUPS pg
;
```

For example, you might want your results to call the PRODUCT_GROUP column "GL_CODE" and the DESCRIPTION column "GL_NAME". This can be done by adding the word AS after the column name, then following that with the Column Alias

Column Aliases

Column Aliases - Optional

```
SELECT
pg.PRODUCT_GROUP as GL_CODE
, pg.DESCRIPTION as GL_NAME
FROM PRODUCT_GROUPS pg
;
```

Each column can have an alias, though they aren't necessary. Use them if it will make your results easier to understand.

Column Aliases

Column Aliases - Results

```
SELECT
pg.PRODUCT_GROUP as GL_CODE
, pg.DESCRIPTION as GL_NAME
FROM PRODUCT_GROUPS pg
;
```

GL_CODE	GL_NAME
412	Cloud_Software_Applications
437	Amazon_Points
414	A_Drive
416	Deal_Sourcer
417	Amazon_Sourced

Column Aliases alter the column headers in your results

Column Aliases

Column Aliases – As Optional, too

```
SELECT
pg.PRODUCT_GROUP GL_CODE
, pg.DESCRIPTION GL_NAME
FROM PRODUCT_GROUPS pg
;
```

Strictly speaking, you don't need the AS in between the column name and the column alias, but it keeps the code easy to read by clearly distinguishing between the two.

Column Aliases

Column Aliases - Underscores

```
SELECT
pg.PRODUCT_GROUP as GL CODE
, pg.DESCRIPTION as GL NAME
FROM PRODUCT_GROUPS pg
;
```

You may have noticed the usage of UNDERSCORES between the words in column aliases. If you try to use spaces, the query will throw an error.

Column Aliases

Column Aliases – No Spaces

```
SELECT
pg.PRODUCT_GROUP as GL CODE
, pg.DESCRIPTION as GL NAME
FROM PRODUCT_GROUPS pg
;
```

You may have noticed the usage of UNDERSCORES between the words in column aliases. If you try to use spaces, the query will error.

Column Aliases

Column Aliases – Unless You Use ""

```
SELECT
pg.PRODUCT_GROUP as "GL CODE"
, pg.DESCRIPTION as "GL NAME"
FROM PRODUCT_GROUPS pg
;
```

If you absolutely want spaces, then put the column alias in double quotes.

Column Aliases

A Quick Note About Quotes

```
SELECT
pg.PRODUCT_GROUP as "GL CODE"
, pg.DESCRIPTION as "GL NAME"
FROM PRODUCT_GROUPS pg
;
```

Column Aliases

" isn't "

```
SELECT
pg.PRODUCT_GROUP as "GL CODE"
, pg.DESCRIPTION as "GL NAME"
FROM PRODUCT_GROUPS pg
;
```

The single and double quote characters that are produced when typing in Microsoft products (Outlook, Excel, Word, etc) are not recognized by Redshift.

Column Aliases

Edit SQL in Notepad

```
SELECT
pg.PRODUCT_GROUP as "GL CODE"
, pg.DESCRIPTION as "GL NAME"
FROM PRODUCT_GROUPS pg
;
```

Use Notepad or (better yet) Notepad++ to edit your SQL to avoid invalid character errors.

Notepad++ is available in Advertised Programs as “Open Source Notepad++”

- Types of Elements

Elements

Other SELECT Clause elements

```
SELECT fcs.WAREHOUSE_ID
, 13
, 'Howdy!'
, fcs.REGION_ID + 5
, SUBSTR(fcs.NAME,0,5)
, ROWNUM
FROM D_WAREHOUSES fcs;
```

You can google about other elements you can put in your SELECT clause, besides simply column names, like literals, functions, and pseudo columns.

Elements

Results Not Sorted

```
SELECT
pg.PRODUCT_GROUP
, pg.DESCRIPTION
FROM PRODUCT_GROUPS pg
;
```

You may have noticed that the results of our query weren't in any particular order.

PRODUCT_GROUP	DESCRIPTION
412	Cloud_Software_Applications
437	Amazon_Points
414	A_Drive
416	Deal_Sourcer
417	Amazon_Sourced
420	Financial_Products
424	Digital_Text_2
425	Digital_Accessories_2
251	Gourmet
241	Watches
236	Misc SDP
234	Travel Store
259	Sports Memorabilia
258	Posters
309	Shoes
23	Electronics
21	Toys

- Concatenation

Concatenation

SQL standard symbol for concatenation is ||

```
SELECT
pg.PRODUCT_GROUP || pg.DESCRIPTION as
glcode_description
FROM PRODUCT_GROUPS pg
;
```

- Order by

ORDER BY

```
SELECT
pg.PRODUCT_GROUP
, pg.DESCRIPTION
FROM PRODUCT_GROUPS pg
ORDER BY
pg.PRODUCT_GROUP
;
```

Specify the column details to sort while adding an ORDER BY clause(after the FROM clause)

ORDER BY

Use Only If Needed

Because the ORDER BY operation has to be run after the rest of the query completes, it adds another costly step that adds time and uses temp space.

As your queries become more complex, ensuring they are as "efficient as possible is of utmost importance.

If you don’t need the ORDER BY clause, don’t include it.

- ETL Tour

ETL Manager Tour

- Getting Started with ETL
- Anatomy of ETL Manager
- Creating & Running a Data Feed
- Job Run Details Page
- Viewing Your Jobs
- Dependency Syntax
- Hoot

Reference link for ETL:
https://w.amazon.com/index.php/DanGSQLClass/IntroToSqlEtl/Lesson1#HLesson1Homework

Lesson 1: Homework

1. Create a query that pulls an alphabetized list of Warehouse IDs from the table D_WAREHOUSES, changing the name of the Warehouse ID column to ‘FC’.
2. Edit the query to add the column REGION_ID, add an element called ‘CALC’ that multiplies the Region ID by 10, add an element called ‘FACTOR’ that is populated with the number 10 for all records, and add an element called FC_REGION that concatenates the WAREHOUSE_ID and the REGION_ID columns with an underscore in between (e.g. PHL1_1).

Here is a description of the D_WAREHOUSES table. We’ll talk more about exploring tables & columns in the future, and what all this information means, but for now, all you need to know is that the list of column names, so you can play around a bit with querying this table using ETL Manager.

Lesson 1: Homework

Table Name: D_WAREHOUSES

Column Name	Data Type	Data Length	Data Precision	Nullable?	Num Distinct
CAN_SHIP_INTERNALLY	CHAR	1		N	2
DB_NAME	VARCHAR2	8		Y	57
DW_CREATION_DATE	DATE	7		N	19
DW_LAST_UPDATED	DATE	7		N	1
HAS_AMAZON_INVENTORY	CHAR	1		N	2
IP_ADDRESS_LIST_ID	NUMBER	22	38	Y	51
IS_DELAYED_ALLOCATION	CHAR	1		N	2
IS_DROPSHIP	CHAR	1		N	2
IS_RETURNS_ONLY	CHAR	1		N	2
NAME	VARCHAR2	50		N	3340
REGION_ID	NUMBER	22		N	3
WAREHOUSE_ID	CHAR	4		N	3453

Remember: One of the great things about SQL is that there are usually several ways to get to the same answer. Different people's minds think about and solve problems in different ways, and you'll likely find some methods that work for you that may be different than what your peers are doing. A good SQL coder is a creative SQL coder, so don't be afraid to try something 'off-book'.

END