PIM Training Program

SQL

Date Function

Learning Objective

At the end of the module, the learner should be able to understand date functionalities in Datanet to effectively write queries using date data type.

Agenda

- SQL
  - DATE Columns
  - The TO_CHAR() Function with Dates
  - The TRUNC() Function with Dates
  - Other date functions
  - Recap of date functions
  - TO_DATE() Function
  - Using BETWEEN with Dates
  - Adding and Subtracting Dates
- ETL
  - Using the Run Date Wildcard
- Lesson 5: Assignment

DATE Columns

DATE Columns

TO_DATE() Function

- Converts a text string into a Date.
- You enter the text string (inside single quotes)
- You indicate the format, telling Redshift which characters are the year, month, day, hour, etc.
- Redshift can then perform activities with date that it can’t perform with a text string (such as adding & subtracting days)

DATE Columns

Lets look at an example explaining the TO_DATE function for Date Columns

D_DAILY_ORDERS

Contains aggregated daily sales information for each ASIN, in each Region Marketplace, for each condition, merchant, etc

. D_DAILY_ORDERS Doesn't have item name.

Partitioned by:

- REGION_ID
- ACTIVITY_DAY

D_DAILY_ORDERS	
REGION_ID [Part:1]	NUMBER(2,0)
LEGAL_ENTITY_ID	NUMBER(38,0)
MARKETPLACE_ID	NUMBER(38,0)
ACTIVITY_DAY [Part:2]	DATE
ASIN	CHAR(10)
MERCHANT_CUSTOMER_ID	NUMBER(38,0)
EXCHANGE_ITEM_CONDITION_ID	NUMBER(2,0)
ORDER_TYPE	VARCHAR2(10)
FREE_REP_REASON_CODE	VARCHAR2(80)
ORDER_METHOD	VARCHAR2(1)
IS_NYP	CHAR(1)
COI_GL_PRODUCT_GROUP	NUMBER(4,0)
BASE_CURRENCY_CODE	VARCHAR2(10)
FULFILLMENT_MANAGER_ID	NUMBER(38,0)
OFFERING_SKU	VARCHAR2(40)
ORDERED_UNITS	NUMBER(38,0)
ORDERED_AMT	NUMBER(38,14)
ADJUSTED_UNITS	NUMBER(38,0)
ADJUSTED_AMT	NUMBER(38,14)
…	
…	
SHIPCHRG_DISC_SAMEDAYCANC_AMT	NUMBER(38,14)
GIFTWRAP_DISC_AMT	NUMBER(38,14)
GIFTWRAP_DISC_ADJUSTED_AMT	NUMBER(38,14)
GIFTWRAP_DISC_SAMEDAYCANC_AMT	NUMBER(38,14)

The datatype of columns like ACTIVITY_DAY is DATE.

DATE Columns

D_MP_ASINS_ESSENTIALS	
REGION_ID [PK:3]	NUMBER(2,0)
MARKETPLACE_ID [PK:2]	NUMBER(38,0)
ASIN [PK:1]	CHAR(10)
BASE_CURRENCY_CODE	VARCHAR2(15)
CATEGORY_CODE	VARCHAR2(96)
GL_PRODUCT_GROUP	NUMBER(10,0)
GL_PRODUCT_GROUP_DESC	VARCHAR2(100)
ITEM_NAM	
MSRP	
MSRP_TAX_	
PRODUCT_AVAILABLE_DAY	DATE
PRODUCT_SITE_LAUNCH_DAY	DATE
PRODUCT_TIER_ID	VARCHAR2(50)
PUBLICATION_DAY	DATE
STREET_DAY	DATE
SUBCATEGORY_CODE	VARCHAR2(96)
DW_CREATION_DATE	DATE
DW_LAST_UPDATED	DATE
IS_DELETED	CHAR(1)
VERSION	NUMBER(38,0)

D_DAILY_ORDERS	
REGION_ID [Part:1]	NUMBER(2,0)
LEGAL_ENTITY_ID	NUMBER(38,0)
MARKETPLACE_ID	NUMBER(38,0)
ACTIVITY_DAY [Part:2]	DATE
ASIN	CHAR(10)
MERCHANT_CUSTOMER_ID	NUMBER(38,0)
EXCHANGE_ITEM_CONDITION_ID	NUMBER(2,0)
	VARCHAR2(10)
	VARCHAR2(80)
	VARCHAR2(1)
	CHAR(1)
[illegible]	NUMBER(4,0)
BASE_CURRENCY_CODE	VARCHAR2(10)
FULFILLMENT_MANAGER_ID	NUMBER(38,0)
OFFERING_SKU	VARCHAR2(40)
ORDERED_UNITS	NUMBER(38,0)
ORDERED_AMT	NUMBER(38,14)
ADJUSTED_UNITS	NUMBER(38,0)
ADJUSTED_AMT	NUMBER(38,14)
…	
…	
SHIPCHRG_DISC_SAMEDAYCANC_AMT	NUMBER(38,14)
GIFTWRAP_DISC_AMT	NUMBER(38,14)
GIFTWRAP_DISC_ADJUSTED_AMT	NUMBER(38,14)
GIFTWRAP_DISC_SAMEDAYCANC_AMT	NUMBER(38,14)

Which US Book ASINs, if any, were sold greater than or equal to 1000 units yesterday?

DATE Columns

D_MP_ASINS_ESSENTIALS	
REGION_ID [PK:3]	NUMBER(2,0)
MARKETPLACE_ID [PK:2]	NUMBER(38,0)
ASIN [PK:1]	CHAR(10)
BASE_CURRENCY_CODE	VARCHAR2(15)
CATEGORY_CODE	VARCHAR2(96)
GL_PRODUCT	
GL_PRODUCT	
ITEM_NAME	
MSRP	
MSRP_TAX_A	
PRODUCT_AV	
PRODUCT_SI	
PRODUCT_TI	
PUBLICATION	
STREET_DAY	
SUBCATEGOR	
DW_CREATIO	
DW_LAST_UF	
IS_DELETED	
VERSION	

D_DAILY_ORDERS	
REGION_ID [Part:1]	NUMBER(2,0)
LEGAL_ENTITY_ID	NUMBER(38,0)
MARKETPLACE_ID	NUMBER(38,0)
ACTIVITY_DAY [Part:2]	DATE
ASIN	CHAR(10)
_CUSTOMER_ID	NUMBER(38,0)
ITEM_CONDITION_ID	NUMBER(2,0)
E	VARCHAR2(10)
EASON_CODE	VARCHAR2(80)
THOD	VARCHAR2(1)
	CHAR(1)
DUCT_GROUP	NUMBER(4,0)
ENCY_CODE	VARCHAR2(10)
T_MANAGER_ID	NUMBER(38,0)
KU	VARCHAR2(40)
NITS	NUMBER(38,0)
MT	NUMBER(38,14)
UNITS	NUMBER(38,0)
AMT	NUMBER(38,14)
DISC_SAMEDAYCANC_AMT	NUMBER(38,14)
DISC_AMT	NUMBER(38,14)
DISC_ADJUSTED_AMT	NUMBER(38,14)
DISC_SAMEDAYCANC_AMT	NUMBER(38,14)

```
SELECT
ddo.ASIN
, dma.ITEM_NAME
, SUM(ddo.ORDERED_UNITS)
FROM D_MP_ASINS_ESSENTIALS dma
JOIN D_DAILY_ORDERS ddo
   ON dma.REGION_ID = ddo.REGION_ID
   AND dma.MARKETPLACE_ID = ddo.MARKETPLACE_ID
   AND dma.ASIN = ddo.ASIN
WHERE dma.REGION_ID = 1
AND dma.MARKETPLACE_ID = 1
AND ddo.REGION_ID = 1
AND ddo.ACTIVITY_DAY = TO_DATE('20110914','YYYYMMDD')
AND ddo.MARKETPLACE_ID = 1
AND dma.GL_PRODUCT_GROUP = 14
GROUP BY
ddo.ASIN
, dma.ITEM_NAME
HAVING SUM(ddo.ORDERED_UNITS) >= 1000
ORDER BY SUM(ddo.ORDERED_UNITS) DESC
;
```

DATE Columns

```
SELECT
ddo.ASIN
, dma.ITEM_NAME
, SUM(ddo.ORDERED_UNITS)
FROM D_MP_ASINS_ESSENTIALS dma
JOIN D_DAILY_ORDERS ddo
   ON dma.REGION_ID = ddo.REGION_ID
   AND dma.MARKETPLACE_ID = ddo.MARKETPLACE_ID
   AND dma.ASIN = ddo.ASIN
WHERE dma.REGION_ID = 1
AND dma.MARKETPLACE_ID = 1
AND ddo.REGION_ID = 1
AND ddo.ACTIVITY_DAY = TO_DATE('20110914','YYYYMMDD')
AND ddo.MARKETPLACE_ID = 1
AND dma.GL_PRODUCT_GROUP = 14
GROUP BY
ddo.ASIN
, dma.ITEM_NAME
HAVING SUM(ddo.ORDERED_UNITS) >= 1000
ORDER BY SUM(ddo.ORDERED_UNITS) DESC
;
```

```
AND ddo.ACTIVITY_DAY = TO_DATE('20110914','YYYYMMDD')
```

Output:

ASIN	ITEM_NAME	SUM(DDO.ORDERED_UNITS)
1401324258	Jacqueline Kennedy: Historic Conversations on Life with John F. Kennedy	4715
0439023521	The Hunger Games	1229
0425245136	The Help (Movie Tie-In)	1216
159562015X	StrengthsFinder 2.0	1121
140020237X	StandOut: The Groundbreaking New Strengths Assessment from the Leader of the Strengths Revolution	1026
0874478529	The Official SAT Study Guide, 2nd edition	1023

DATE columns

```
AND ddo.ACTIVITY_DAY = TO_DATE('20110914','YYYYMMDD')
```

```
TO_DATE('20110914','YYYYMMDD')
```

DATE columns

AND ddo.ACTIVITY_DAY = TO_DATE('20110914','YYYYMMDD')

TO_DATE('20110914','YYYYMMDD')

2011 = YYYY

09 = MM

14 = DD

Therefore Date is Sept 14, 2011

DATE columns

```
AND ddo.ACTIVITY_DAY = TO_DATE('20110914','YYYY/MM/DD')
```

```
TO_DATE('20110914','YYYY/MM/DD')
```

Make sure the format you enter EXACTLY matches the format of the text string you entered or you'll get an error like:

ORA-01843: not a valid month

DATE columns

But why do we need TO_DATE?

<u>Bottom Line:</u>

ddo.ACTIVITY_DAY = TO_DATE('20110914','YYYYMMDD')

WORKS

ddo.ACTIVITY_DAY = '20110914'

DOESN’T

DATE columns

<u>Operators available for Date Functions</u>

Equal To	=
Greater Than or Equal To	>=
Less Than or Equal To	<=
Greater Than	>
Less Than	<
BETWEEN	

DATE columns

Difference between “=“ and “BETWEEN”

1

```
AND ddo.ACTIVITY_DAY = TO_DATE('20110914','YYYYMMDD')
```

Returns only records where ACTIVITY_DAY is equal to 9/14/2011

2

```
AND ddo.ACTIVITY_DAY BETWEEN
            TO_DATE('20110914','YYYYMMDD')
   AND TO_DATE('20110920','YYYYMMDD')
```

Returns all records where ACTIVITY_DAY is Greater Than or Equal To 9/14/2011 and Less Than or Equal To 9/20/2011

DATE columns

```
SELECT
ddo.ASIN
, dma.ITEM_NAME
, SUM(ddo.ORDERED_UNITS)
FROM D_MP_ASINS_ESSENTIALS dma
JOIN D_DAILY_ORDERS ddo
   ON dma.REGION_ID = ddo.REGION_ID
   AND dma.MARKETPLACE_ID = ddo.MARKETPLACE_ID
   AND dma.ASIN = ddo.ASIN
WHERE dma.REGION_ID = 1
AND dma.MARKETPLACE_ID = 1
AND ddo.REGION_ID = 1
AND ddo.ACTIVITY_DAY BETWEEN
          TO_DATE('20110914','YYYYMMDD')
    AND TO_DATE('20110920','YYYYMMDD')
AND ddo.MARKETPLACE_ID = 1
AND dma.GL_PRODUCT_GROUP = 14
GROUP BY
ddo.ASIN
, dma.ITEM_NAME
HAVING SUM(ddo.ORDERED_UNITS) >= 1000
ORDER BY SUM(ddo.ORDERED_UNITS) DESC
;
```

DATE columns

Run Date Wildcards

Three to Choose From:

{RUN_DATE_YYYY/MM/DD}

{RUN_DATE_YYYY-MM-DD}

{RUN_DATE_YYYYMMDD}

https://w.amazon.com/index.php/ETLWildcards

DATE columns

Here is an example for {RUN_DATE_YYYYMMDD}

Returns a Text String in the format YYYYMMDD that matches the Date you select when you run the job (or the dataset date of a scheduled job).

20110921

Therefore, you have to treat it just like a text string

DATE columns

Example of Run date wildcard using between operator

```
AND ddo.ACTIVITY_DAY BETWEEN
    TO_DATE('20110914','YYYYMMDD')
AND TO_DATE('{RUN_DATE_YYYYMMDD}','YYYYMMDD')
```

Returns all dates Greater Than or Equal To 9/14/2011 and Less Than or Equal To the Run Date

If Run Date is 9/20/2011, then 9/14/2011 thru 9/20/2011

If Run Date is 9/21/2011, then 9/14/2011 thru 9/21/2011

What date range would it return today if I scheduled it to run daily?

DATE columns

```
AND ddo.ACTIVITY_DAY BETWEEN
        TO_DATE('20110914','YYYYMMDD')
AND TO_DATE('{RUN_DATE_YYYYMMDD}','YYYYMMDD')
```

Notice that:

- You still need the TO_DATE() function
- You still need single quotes around it
- You still need to match the format

DATE columns

```
SELECT
ddo.ASIN
, dma.ITEM_NAME
, SUM(ddo.ORDERED_UNITS)
FROM D_MP_ASINS_ESSENTIALS dma
JOIN D_DAILY_ORDERS ddo
   ON dma.REGION_ID = ddo.REGION_ID
   AND dma.MARKETPLACE_ID = ddo.MARKETPLACE_ID
   AND dma.ASIN = ddo.ASIN
WHERE dma.REGION_ID = 1
AND dma.MARKETPLACE_ID = 1
AND ddo.REGION_ID = 1
AND ddo.ACTIVITY_DAY BETWEEN
          TO_DATE('20110914','YYYYMMDD')
    AND TO_DATE('{RUN_DATE_YYYYMMDD}','YYYYMMDD')
AND ddo.MARKETPLACE_ID = 1
AND dma.GL_PRODUCT_GROUP = 14
GROUP BY
ddo.ASIN
, dma.ITEM_NAME
HAVING SUM(ddo.ORDERED_UNITS) >= 1000
ORDER BY SUM(ddo.ORDERED_UNITS) DESC
;
```

DATE columns

ONLY WHEN YOUR QUERY INCLUDES A RUN DATE WILDCARD DOES THE RUN DATE YOU SELECT MATTER

If you didn’t use a Run Date Wildcard, select previous day’s date

DATE columns

Getting Dynamic with Dates - Adding and Subtracting Dates

You can add and subtract days from dates

You can add and subtract from DATEs

- This is useful for setting date ranges
- Values:
    - 1 = day (base case)
    - 2/24 = two hours
    - 45/1440 = forty five minutes

DATE columns

```
AND ddo.ACTIVITY_DAY BETWEEN
	TO_DATE('{RUN_DATE_YYYYMMDD}','YYYYMMDD')-6
AND TO_DATE('{RUN_DATE_YYYYMMDD}','YYYYMMDD')
```

If Run Date is 9/17/2011, then 9/11/2011 thru 9/17/2011

Note that -6 returns a 7 day window, since BETWEEN is inclusive.

This is how to set up a query that dynamically returns a week of data. Schedule it to run on Sunday, so Run Date is Saturday, and you get last week.

DATE columns

```
SELECT
ddo.ACTIVITY_DAY
, SUM(ddo.ORDERED_UNITS)
FROM D_MP_ASINS_ESSENTIALS dma
JOIN D_DAILY_ORDERS ddo
   ON dma.REGION_ID = ddo.REGION_ID
   AND dma.MARKETPLACE_ID = ddo.MARKETPLACE_ID
   AND dma.ASIN = ddo.ASIN
WHERE dma.REGION_ID = 1
AND dma.MARKETPLACE_ID = 1
AND ddo.REGION_ID = 1
AND ddo.ACTIVITY_DAY BETWEEN
 TO_DATE('{RUN_DATE_YYYYMMDD}','YYYYMMDD')-6
    AND TO_DATE('{RUN_DATE_YYYYMMDD}','YYYYMMDD')
AND ddo.MARKETPLACE_ID = 1
AND dma.GL_PRODUCT_GROUP = 14
GROUP BY
ddo.ACTIVITY_DAY
;
```

So we got Daily data for the range specified

ACTIVITY_DAY	SUM(DDO.ORDERED_UNITS)
11-Sep-11	691542
12-Sep-11	942830
13-Sep-11	918070
14-Sep-11	883751
15-Sep-11	819736
16-Sep-11	698795
17-Sep-11	529303

DATE columns

```
SELECT
ddo.ACTIVITY_DAY
, SUM(ddo.ORDERED_UNITS)
FROM D_MP_ASINS_ESSENTIALS dma
JOIN D_DAILY_ORDERS ddo
   ON dma.REGION_ID = ddo.REGION_ID
   AND dma.MARKETPLACE_ID = ddo.MARKETPLACE_ID
   AND dma.ASIN = ddo.ASIN
WHERE dma.REGION_ID = 1
AND dma.MARKETPLACE_ID = 1
AND ddo.REGION_ID = 1
AND ddo.ACTIVITY_DAY BETWEEN
 TO_DATE('{RUN_DATE_YYYYMMDD}','YYYYMMDD')-6
    AND TO_DATE('{RUN_DATE_YYYYMMDD}','YYYYMMDD')
AND ddo.MARKETPLACE_ID = 1
AND dma.GL_PRODUCT_GROUP = 14
GROUP BY
ddo.ACTIVITY_DAY
;
```

Note the date format

ACTIVITY_DAY	SUM(DDO.ORDERED_UNITS)
11-Sep-11	691542
12-Sep-11	942830
13-Sep-11	918070
14-Sep-11	883751
15-Sep-11	819736
16-Sep-11	698795
17-Sep-11	529303

DATE columns

```
SELECT
ddo.ACTIVITY_DAY
, SUM(ddo.ORDERED_UNITS)
FROM D_MP_ASINS_ESSENTIALS dma
JOIN D_DAILY_ORDERS ddo
   ON dma.REGION_ID = ddo.REGION_ID
   AND dma.MARKETPLACE_ID = ddo.MARKETPLACE_ID
   AND dma.ASIN = ddo.ASIN
WHERE dma.REGION_ID = 1
AND dma.MARKETPLACE_ID = 1
AND ddo.REGION_ID = 1
AND ddo.ACTIVITY_DAY BETWEEN
 TO_DATE('{RUN_DATE_YYYYMMDD}','YYYYMMDD')-6
    AND TO_DATE('{RUN_DATE_YYYYMMDD}','YYYYMMDD')
AND ddo.MARKETPLACE_ID = 1
AND dma.GL_PRODUCT_GROUP = 14
GROUP BY
ddo.ACTIVITY_DAY
;
```

We know the last date is Sept 17, 2011, but it could also be Sept 11, 2017

ACTIVITY_DAY	SUM(DDO.ORDERED_UNITS)
11-Sep-11	691542
12-Sep-11	942830
13-Sep-11	918070
14-Sep-11	883751
15-Sep-11	819736
16-Sep-11	698795
17-Sep-11	529303

DATE columns

The Same Date can be written many different ways

9/17/2011	9/17/11
09/17/2011	09/17/11
20110917	2011-09-17
17/09/2011	9-17-11
17-Sept-11	9-17-2011

September 17th, 2011

DATE columns

The Same Date can be written in different ways

So it’s a good idea to force your results into the format that makes sense to your customer, to avoid any potential misinterpretation

(This also helps Excel avoid false assumptions, like assuming a Publication Date of 9/17/11 means 9/17/2011, not 9/17/1911)

- TO_CHAR() Function with Dates

TO_CHAR() Function with Dates

- Converts a Date into a Character string.
- Opposite of TO_DATE()
- You enter the Date column
- You indicate the format in which you want it to return, telling Redshift which characters are the year, month, day, hour, etc.

TO_CHAR() Function with Dates

```
TO_CHAR(ddo.ACTIVITY_DAY, 'MM/DD/YYYY')
```

Forces to return the text string 09/17/2011

TO_CHAR() Function with Dates

```
SELECT
TO_CHAR(ddo.ACTIVITY_DAY, 'MM/DD/YYYY')
, SUM(ddo.ORDERED_UNITS)
FROM D_MP_ASINS_ESSENTIALS dma
JOIN D_DAILY_ORDERS ddo
   ON dma.REGION_ID = ddo.REGION_ID
   AND dma.MARKETPLACE_ID = ddo.MARKETPLACE_ID
   AND dma.ASIN = ddo.ASIN
WHERE dma.REGION_ID = 1
AND dma.MARKETPLACE_ID = 1
AND ddo.REGION_ID = 1
AND ddo.ACTIVITY_DAY BETWEEN
 TO_DATE('{RUN_DATE_YYYYMMDD}','YYYYMMDD')-6
    AND TO_DATE('{RUN_DATE_YYYYMMDD}','YYYYMMDD')
AND ddo.MARKETPLACE_ID = 1
AND dma.GL_PRODUCT_GROUP = 14
GROUP BY
TO_CHAR(ddo.ACTIVITY_DAY, 'MM/DD/YYYY')
;
```

TO_CHAR() Function with Dates

```
SELECT
TO_CHAR(ddo.ACTIVITY_DAY, 'MM/DD/YYYY')
, SUM(ddo.ORDERED_UNITS)
FROM D_MP_ASINS_ESSENTIALS dma
JOIN D_DAILY_ORDERS ddo
   ON dma.REGION_ID = ddo.REGION_ID
   AND dma.MARKETPLACE_ID = ddo.MARKETPLACE_ID
   AND dma.ASIN = ddo.ASIN
WHERE dma.REGION_ID = 1
AND dma.MARKETPLACE_ID = 1
AND ddo.REGION_ID = 1
AND ddo.ACTIVITY_DAY BETWEEN
 TO_DATE('{RUN_DATE_YYYYMMDD}','YYYYMMDD')-6
    AND TO_DATE('{RUN_DATE_YYYYMMDD}','YYYYMMDD')
AND ddo.MARKETPLACE_ID = 1
AND dma.GL_PRODUCT_GROUP = 14
GROUP BY
TO_CHAR(ddo.ACTIVITY_DAY, 'MM/DD/YYYY')
;
```

Output

```
ACTIVITY_DAY   SUM(DDO.ORDERED_UNITS)
  09/11/2011                   691542
  09/12/2011                   942830
  09/13/2011                   918070
  09/14/2011                   883751
  09/15/2011                   819736
  09/16/2011                   698795
  09/17/2011                   529303
```

TO_CHAR() Function with Dates

```
SELECT
TO_CHAR(ddo.ACTIVITY_DAY, 'MM/DD/YYYY')
, SUM(ddo.ORDERED_UNITS)
FROM D_MP_ASINS_ESSENTIALS dma
JOIN D_DAILY_ORDERS ddo
   ON dma.REGION_ID = ddo.REGION_ID
   AND dma.MARKETPLACE_ID = ddo.MARKETPLACE_ID
   AND dma.ASIN = ddo.ASIN
WHERE dma.REGION_ID = 1
AND dma.MARKETPLACE_ID = 1
AND ddo.REGION_ID = 1
AND ddo.ACTIVITY_DAY BETWEEN
 TO_DATE('{RUN_DATE_YYYYMMDD}','YYYYMMDD')-6
    AND TO_DATE('{RUN_DATE_YYYYMMDD}','YYYYMMDD')
AND ddo.MARKETPLACE_ID = 1
AND dma.GL_PRODUCT_GROUP = 14
GROUP BY
TO_CHAR(ddo.ACTIVITY_DAY, 'MM/DD/YYYY')
;
```

REVIEW: How would we pull 2 weeks of data?

ACTIVITY_DAY	SUM(DDO.ORDERED_UNITS)
09/11/2011	691542
09/12/2011	942830
09/13/2011	918070
09/14/2011	883751
09/15/2011	819736
09/16/2011	698795
09/17/2011	529303

TO_CHAR() Function with Dates

```
SELECT
TO_CHAR(ddo.ACTIVITY_DAY, 'MM/DD/YYYY')
, SUM(ddo.ORDERED_UNITS)
FROM D_MP_ASINS_ESSENTIALS dma
JOIN D_DAILY_ORDERS ddo
   ON dma.REGION_ID = ddo.REGION_ID
   AND dma.MARKETPLACE_ID = ddo.MARKETPLACE_ID
   AND dma.ASIN = ddo.ASIN
WHERE dma.REGION_ID = 1
AND dma.MARKETPLACE_ID = 1
AND ddo.REGION_ID = 1
AND ddo.ACTIVITY_DAY BETWEEN
 TO_DATE('{RUN_DATE_YYYYMMDD}','YYYYMMDD')-13
    AND TO_DATE('{RUN_DATE_YYYYMMDD}','YYYYMMDD')
AND ddo.MARKETPLACE_ID = 1
AND dma.GL_PRODUCT_GROUP = 14
GROUP BY
TO_CHAR(ddo.ACTIVITY_DAY, 'MM/DD/YYYY')
;
```

ACTIVITY_DAY	SUM(DDO.ORDERED_UNITS)
09/4/2011	633132
09/5/2011	860042
09/6/2011	1077841
09/7/2011	1057861
09/8/2011	939424
09/9/2011	774693
09/10/2011	579284
09/11/2011	691542
09/12/2011	942830
09/13/2011	918070
09/14/2011	883751
09/15/2011	819736
09/16/2011	698795
09/17/2011	529303

TO_CHAR() Function with Dates

```
SELECT
TO_CHAR(ddo.ACTIVITY_DAY, 'MM/DD/YYYY')
, SUM(ddo.ORDERED_UNITS)
FROM D_MP_ASINS_ESSENTIALS dma
JOIN D_DAILY_ORDERS ddo
   ON dma.REGION_ID = ddo.REGION_ID
   AND dma.MARKETPLACE_ID = ddo.MARKETPLACE_ID
   AND dma.ASIN = ddo.ASIN
WHERE dma.REGION_ID = 1
AND dma.MARKETPLACE_ID = 1
AND ddo.REGION_ID = 1
AND ddo.ACTIVITY_DAY BETWEEN
 TO_DATE('{RUN_DATE_YYYYMMDD}','YYYYMMDD')-13
    AND TO_DATE('{RUN_DATE_YYYYMMDD}','YYYYMMDD')
AND ddo.MARKETPLACE_ID = 1
AND dma.GL_PRODUCT_GROUP = 14
GROUP BY
TO_CHAR(ddo.ACTIVITY_DAY, 'MM/DD/YYYY')
;
```

Now, what if we want 2 weeks of data, but by week not by day?

ACTIVITY_DAY	SUM(DDO.ORDERED_UNITS)
09/4/2011	633132
09/5/2011	860042
09/6/2011	1077841
09/7/2011	1057861
09/8/2011	939424
09/9/2011	774693
09/10/2011	579284
09/11/2011	691542
09/12/2011	942830
09/13/2011	918070
09/14/2011	883751
09/15/2011	819736
09/16/2011	698795
09/17/2011	529303

- TRUNC() Function with Dates

TRUNC() Function with Dates

- Function truncates date to the level of specificity the user chooses

Truncates a date to the start of the period you indicate, but keeps it as a Date (not text)

Default is the start of the Day

Can be used to change a date to the Start of a Week, Month, Quarter, Year, Century, Epoch, etc.

TRUNC() Function with Dates

Common Formats:

DD = Day

D = Week

MM = Month

Q = Quarter

Y = Year

Example of output for D and DD

D = first second of first hour of first day of week

DDD = first second of first hour of same day

TRUNC() Function with Dates

```
TRUNC(ddo.ACTIVITY_DAY, 'D')
```

If ACTIVITY_DAY is 9/17/2011, it returns 9/11/2011
If ACTIVITY_DAY is 9/16/2011, it returns 9/11/2011
If ACTIVITY_DAY is 9/15/2011, it returns 9/11/2011
If ACTIVITY_DAY is 9/14/2011, it returns 9/11/2011
If ACTIVITY_DAY is 9/13/2011, it returns 9/11/2011
If ACTIVITY_DAY is 9/12/2011, it returns 9/11/2011
If ACTIVITY_DAY is 9/11/2011, it returns 9/11/2011
If ACTIVITY_DAY is 9/10/2011, it returns 9/04/2011
If ACTIVITY_DAY is 9/17/2011, it returns 9/04/2011
etc

TRUNC() Function with Dates

```
SELECT
TRUNC(ddo.ACTIVITY_DAY, 'D') as WEEK
, SUM(ddo.ORDERED_UNITS)
FROM D_MP_ASINS_ESSENTIALS dma
JOIN D_DAILY_ORDERS ddo
   ON dma.REGION_ID = ddo.REGION_ID
   AND dma.MARKETPLACE_ID = ddo.MARKETPLACE_ID
   AND dma.ASIN = ddo.ASIN
WHERE dma.REGION_ID = 1
AND dma.MARKETPLACE_ID = 1
AND ddo.REGION_ID = 1
AND ddo.ACTIVITY_DAY BETWEEN
 TO_DATE('{RUN_DATE_YYYYMMDD}','YYYYMMDD')-13
    AND TO_DATE('{RUN_DATE_YYYYMMDD}','YYYYMMDD')
AND ddo.MARKETPLACE_ID = 1
AND dma.GL_PRODUCT_GROUP = 14
GROUP BY
TRUNC(ddo.ACTIVITY_DAY, 'D')
;
```

TRUNC() Function with Dates

```
SELECT
TRUNC(ddo.ACTIVITY_DAY, 'D') as WEEK
, SUM(ddo.ORDERED_UNITS)
FROM D_MP_ASINS_ESSENTIALS dma
JOIN D_DAILY_ORDERS ddo
   ON dma.REGION_ID = ddo.REGION_ID
   AND dma.MARKETPLACE_ID = ddo.MARKETPLACE_ID
   AND dma.ASIN = ddo.ASIN
WHERE dma.REGION_ID = 1
AND dma.MARKETPLACE_ID = 1
AND ddo.REGION_ID = 1
AND ddo.ACTIVITY_DAY BETWEEN
 TO_DATE('{RUN_DATE_YYYYMMDD}','YYYYMMDD')-13
    AND TO_DATE('{RUN_DATE_YYYYMMDD}','YYYYMMDD')
AND ddo.MARKETPLACE_ID = 1
AND dma.GL_PRODUCT_GROUP = 14
GROUP BY
TRUNC(ddo.ACTIVITY_DAY, 'D')
;
```

(We have intentionally removed the TO_CHAR() function, to avoid confusion.)

TRUNC() Function with Dates

```
SELECT
TRUNC(ddo.ACTIVITY_DAY, 'D') as WEEK
, SUM(ddo.ORDERED_UNITS)
FROM D_MP_ASINS_ESSENTIALS dma
JOIN D_DAILY_ORDERS ddo
   ON dma.REGION_ID = ddo.REGION_ID
   AND dma.MARKETPLACE_ID = ddo.MARKETPLACE_ID
   AND dma.ASIN = ddo.ASIN
WHERE dma.REGION_ID = 1
AND dma.MARKETPLACE_ID = 1
AND ddo.REGION_ID = 1
AND ddo.ACTIVITY_DAY BETWEEN
 TO_DATE('{RUN_DATE_YYYYMMDD}','YYYYMMDD')-13
    AND TO_DATE('{RUN_DATE_YYYYMMDD}','YYYYMMDD')
AND ddo.MARKETPLACE_ID = 1
AND dma.GL_PRODUCT_GROUP = 14
GROUP BY
TRUNC(ddo.ACTIVITY_DAY, 'D')
;
```

WEEK	SUM(DDO.ORDERED_UNITS)
4-Sep-11	5922277
11-Sep-11	5484027

TRUNC() Function with Dates

```
SELECT
TRUNC(ddo.ACTIVITY_DAY, 'D') as WEEK
, SUM(ddo.ORDERED_UNITS)
FROM D_MP_ASINS_ESSENTIALS dma
JOIN D_DAILY_ORDERS ddo
   ON dma.REGION_ID = ddo.REGION_ID
   AND dma.MARKETPLACE_ID = ddo.MARKETPLACE_ID
   AND dma.ASIN = ddo.ASIN
WHERE dma.REGION_ID = 1
AND dma.MARKETPLACE_ID = 1
AND ddo.REGION_ID = 1
AND ddo.ACTIVITY_DAY BETWEEN
 TO_DATE('{RUN_DATE_YYYYMMDD}','YYYYMMDD')-13
    AND TO_DATE('{RUN_DATE_YYYYMMDD}','YYYYMMDD')
AND ddo.MARKETPLACE_ID = 1
AND dma.GL_PRODUCT_GROUP = 14
GROUP BY
TRUNC(ddo.ACTIVITY_DAY, 'D')
;
```

Since we removed the TO_CHAR function, are dates are not in standard format again

WEEK	SUM(DDO.ORDERED_UNITS)
4-Sep-11	5922277
11-Sep-11	5484027

TRUNC() Function with Dates

Rolling it all Together

```
TO_CHAR(TRUNC(ddo.ACTIVITY_DAY, 'D'), 'MM/DD/YYYY')
```

TRUNC() Function with Dates

```
SELECT
TO_CHAR(TRUNC(ddo.ACTIVITY_DAY, 'D'), 'MM/DD/YYYY') as WEEK
, SUM(ddo.ORDERED_UNITS)
FROM D_MP_ASINS_ESSENTIALS dma
JOIN D_DAILY_ORDERS ddo
   ON dma.REGION_ID = ddo.REGION_ID
   AND dma.MARKETPLACE_ID = ddo.MARKETPLACE_ID
   AND dma.ASIN = ddo.ASIN
WHERE dma.REGION_ID = 1
AND dma.MARKETPLACE_ID = 1
AND ddo.REGION_ID = 1
AND ddo.ACTIVITY_DAY BETWEEN
 TO_DATE('{RUN_DATE_YYYYMMDD}','YYYYMMDD')-13
    AND TO_DATE('{RUN_DATE_YYYYMMDD}','YYYYMMDD')
AND ddo.MARKETPLACE_ID = 1
AND dma.GL_PRODUCT_GROUP = 14
GROUP BY
TO_CHAR(TRUNC(ddo.ACTIVITY_DAY, 'D'), 'MM/DD/YYYY')
;
```

TRUNC() Function with Dates

```
SELECT
TO_CHAR(TRUNC(ddo.ACTIVITY_DAY, 'D'), 'MM/DD/YYYY') as WEEK
, SUM(ddo.ORDERED_UNITS)
FROM D_MP_ASINS_ESSENTIALS dma
JOIN D_DAILY_ORDERS ddo
   ON dma.REGION_ID = ddo.REGION_ID
   AND dma.MARKETPLACE_ID = ddo.MARKETPLACE_ID
   AND dma.ASIN = ddo.ASIN
WHERE dma.REGION_ID = 1
AND dma.MARKETPLACE_ID = 1
AND ddo.REGION_ID = 1
AND ddo.ACTIVITY_DAY BETWEEN
 TO_DATE('{RUN_DATE_YYYYMMDD}','YYYYMMDD')-13
    AND TO_DATE('{RUN_DATE_YYYYMMDD}','YYYYMMDD')
AND ddo.MARKETPLACE_ID = 1
AND dma.GL_PRODUCT_GROUP = 14
GROUP BY
TO_CHAR(TRUNC(ddo.ACTIVITY_DAY, 'D'), 'MM/DD/YYYY')
;
```

Output:

```
WEEK         SUM(DDO.ORDERED_UNITS)
  09/04/2011                5922277
  09/11/2011                5484027
```

➢Other Date Functions

Points to remember while working on Tables

The various formats in which dates are stored in Amazon tables are:

- Some DATE fields store just the date (9/14/2011)
- Some store the entire datetime (9/14/2011 22:52:30)
- Be careful you know which is which, as 9/14/2011 is not equal to 9/14/2011 22:52:30
- Read more about this in your manual.

Points to remember while working on Tables

SNAPSHOTS vs. ACTIVITY DATES

Some tables store transactional data, like customer orders or shipments.

Some tables store snapshots of what things looked like at the end of each day, like inventory or open Pos.

Other tables aren’t date specific (like D_MP_ASINS_ESSENTIALS)

Other Date Functions

A Closing note on other Date Options

- You may run across queries that use two other date ‘wildcards’: SYSDATE and CURRENT_DATE

- It’s important to remember these return the date the query is running, not the run date of the query

Other Date Functions

A Closing note on other Date Options

Imagine you are running a query on 9/21/2011, but for a run date of 9/17/2011. For example, perhaps you need to rerun a query that errored, but was scheduled to run on Sunday.

```
TO_DATE('{RUN_DATE_YYYYMMDD}','YYYYMMDD') = 9/17/2011
                        TO_DATE(CURRENT_DATE) = 9/21/2011
                                     SYSDATE = 9/21/2011
```

Other Date Functions

A Closing note on other Date Options

This could obviously cause very different results to return, and has the risk of misstating something.

```
TO_DATE('{RUN_DATE_YYYYMMDD}','YYYYMMDD') = 9/17/2011
                 TO_DATE(CURRENT_DATE) = 9/21/2011
                              SYSDATE = 9/21/2011
```

Other Date Functions

A Closing note on other Date Options

This could obviously cause very different results to return, and has the risk of misstating something.

For this reason, it is recommend using the RUN_DATE wildcard.

```
TO_DATE('{RUN_DATE_YYYYMMDD}','YYYYMMDD') = 9/17/2011
                   TO_DATE(CURRENT_DATE) = 9/21/2011
                                 SYSDATE = 9/21/2011
```

- Recap of Date Functions

Recap of Date functions

DATE Columns

- Similarities
  - Of data type ‘DATE’
  - Store full date & time information
- Differences
  - DATE: Truncates to the nearest day
  - DATETIME: Contains complete timestamp in addition to day

DATE	DATETIME
04/08/15	04/08/15 16:23:42

- How do I tell if a column in ETL is DATE or DATETIME?

Recap of Date functions

TO_CHAR() Function

- Core functionality is to translate a numerical date to a string
- Why do we use it? Converting to a more understandable terms.
- Implementation:
  - Part I: column to convert
  - Part II: format to convert to

```
SELECT
ddo.ORDER_ID
, ddo.ORDER_DATETIME
, TO_CHAR(ddo.ORDER_DATETIME,'YYYYMMDD'),
TO_CHAR(ddo.ORDER_DATETIME,'D'),
TO_CHAR(ddo.ORDER_DATETIME,'DAY'),
TO_CHAR(ddo.ORDER_DATETIME,'CC'),
TO_CHAR(ddo.ORDER_DATETIME,'HH AM" on a "Day", the
"DDDTH" day of "YYYY"')
FROM D_DISTRIBUTOR_ORDERS ddo
WHERE ddo.REGION_ID = 1
AND ddo.ORDER_ID = 'M5969483';
```

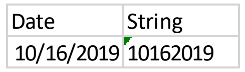
ORDER_ID	ORDER_DATETIME	YYYYMMDD	D	DAY	CC	HH AM" on a "Day", the "DDDTH" day of "YYYY"
M596948 3	3/25/2009 11:14	20090325	4	WEDNESDAY	21	11 AM on a Wednesday, the 084TH day of 2009

Recap of Date functions

Limitations of using TO_CHAR

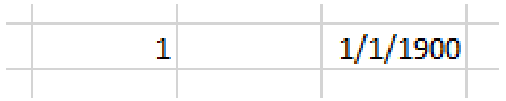
- Once converted from a value to a string, the date is no longer pliable

Example :



Date	String
10/16/2019	10162019

- We like to manipulate dates in SQL, like looking back a week to create weekly date sets
- Think about what happen if you accidentally set the number 1 to a date in Excel:



1	1/1/1900

- Outputs in string format may still be readable by Excel, if the program can interpret the selected format

Recap of Date functions

TRUNC() Function

- Function truncates date to the level of specificity the user chooses
- Specificity dictated by operators
  - D = first second of first hour of first day of week
  - DDD = first second of first hour of same day
- Advantage: preserves 'DATE' data type
- Implementation:

```
SELECT
ddo.ORDER_ID
, ddo.ORDER_DATETIME
, TRUNC(ddo.ORDER_DATETIME,'D')
FROM D_DISTRIBUTOR_ORDERS ddo
WHERE ddo.REGION_ID = 1
AND ddo.ORDER_ID = 'M5969483';
```

ORDER_ID	ORDER_DATETIME	TRUNC(DDO.ORDER_DATETIME,'D')
M5969483	3/25/2009 11:14	3/22/2009

- TO_DATE() Function

TO_DATE() Function

- The opposite of the TO_CHAR function, if a number can be converted back into a date format it translates it
- Commonly used in the WHERE clause to limit results
- Implementation:
  - Part I: date string value
  - Part II: interpretation key (how to read it)

```sql
SELECT
ddo.ORDER_ID
, ddo.ORDER_DAY
FROM D_DISTRIBUTOR_ORDERS ddo
WHERE ddo.REGION_ID = 1
AND ddo.LEGAL_ENTITY_ID = 101
AND ddo.DISTRIBUTOR_ID = 'RANDO'
AND ddo.ORDER_DAY =
TO_DATE('20090325', 'YYYYMMDD');
```

Part I Part II

ORDER_ID	ORDER_DAY
P0618301	3/25/2009
M5969483	3/25/2009

- BETWEEN TO_DATE()

BETWEEN TO_DATE()

- Sets a date range using the TO_DATE() function
- IMPORTANT: be careful if using DATETIME values in this function, as the appended seconds may be left out of your date range if not properly included.
- Implementation:

ORDER_ID	ORDER_DAY
M9119427	3/23/2009
M2666981	3/23/2009
U3517863	3/23/2009
R5273263	3/23/2009
N5183001	3/23/2009
T0475345	3/23/2009
M5969483	3/25/2009
P0618301	3/25/2009

```
SELECT
ddo.ORDER_ID
, ddo.ORDER_DAY
FROM D_DISTRIBUTOR_ORDERS ddo
WHERE ddo.REGION_ID = 1
AND ddo.LEGAL_ENTITY_ID = 101
AND ddo.DISTRIBUTOR_ID = 'RANDO'
AND ddo.ORDER_DAY
BETWEEN
TO_DATE('20090323','YYYYMMDD')
AND TO_DATE('20090325','YYYYMMDD');
```

- Adding and Subtracting dates

Adding and Subtracting dates

- You can add and subtract from DATEs
- This is useful for setting date ranges
- Values:
  - 1 = day (base case)
  - 2/24 = two hours
  - 45/1440 = forty five minutes
- Implementation:

```sql
SELECT
ddo.ORDER_ID
, ddo.ORDER_DATETIME
, ddo.ORDER_DATETIME + 1
FROM D_DISTRIBUTOR_ORDERS ddo
WHERE ddo.REGION_ID = 1
AND ddo.LEGAL_ENTITY_ID = 101
AND ddo.DISTRIBUTOR_ID = 'RANDO'
AND TRUNC(ddo.ORDER_DATETIME)
BETWEEN
TO_DATE('20090323','YYYYMMDD')
AND TO_DATE('20090325','YYYYMMDD');
```

➢ETL Using the Run Date Wildcard

ETL Using the Run Date Wildcard

- ETL accepts wildcards, which are dynamic fields that replace any of the date fields covered in the previous slides.
- Date wildcard: {RUN_DATE_YYYYMMDD}
- By replacing a date with a wildcard you tell ETL to execute the query, inserting the rundate into the SQL operation

```sql
SELECT
ddo.ORDER_ID
, ddo.ORDER_DATETIME
FROM D_DISTRIBUTOR_ORDERS ddo
WHERE ddo.REGION_ID = 1
AND ddo.LEGAL_ENTITY_ID = 101
AND ddo.DISTRIBUTOR_ID = 'RANDO'
AND TRUNC(ddo.ORDER_DATETIME)
BETWEEN TO_DATE('20090325','YYYYMMDD')-2
AND TO_DATE('20090325','YYYYMMDD');
```

```sql
SELECT
ddo.ORDER_ID
, ddo.ORDER_DATETIME
FROM D_DISTRIBUTOR_ORDERS ddo
WHERE ddo.REGION_ID = 1
AND ddo.LEGAL_ENTITY_ID = 101
AND ddo.DISTRIBUTOR_ID = 'RANDO'AND
TRUNC(ddo.ORDER_DATETIME)
BETWEEN TO_DATE('{RUN_DATE_YYYYMMDD}','YYYYMMDD')-2
AND TO_DATE('{RUN_DATE_YYYYMMDD}','YYYYMMDD');
```

ETL Using the Run Date Wildcard

Common uses of wildcard

Yesterday:

```
WHERE ddo.ORDER_DAY = TO_DATE('{RUN_DATE_YYYYMMDD}','YYYYMMDD')
```

Last Week (when scheduled to run on a Sunday, for the Run Date of the last day of last week):

```
WHERE ddo.ORDER_DAY BETWEEN TO_DATE('{RUN_DATE_YYYYMMDD}','YYYYMMDD')-6
                    AND TO_DATE('{RUN_DATE_YYYYMMDD}','YYYYMMDD')
```

Last Month (when scheduled to run on the first of the month, for the Run Date of the last day of the month):

```
WHERE ddo.ORDER_DAY BETWEEN TRUNC(TO_DATE('{RUN_DATE_YYYYMMDD}','YYYYMMDD'),'MM')
                    AND LAST_DAY(TO_DATE('{RUN_DATE_YYYYMMDD}','YYYYMMDD'))
```

Year to Date:

```
WHERE ddo.ORDER_DAY BETWEEN TRUNC(TO_DATE('{RUN_DATE_YYYYMMDD}','YYYYMMDD'),'Y')
                    AND TO_DATE('{RUN_DATE_YYYYMMDD}','YYYYMMDD')
```

- Lesson 5 Assignment

➢ Lesson 5: Assignment

1. Write a query to determine on what date the record in D_WAREHOUSES for the FC PHL1 was created. Use the TO_CHAR function to ensure the date is returned in the format MM/DD/YYYY.
2. Edit the query to determine what the first and last days of the week that record was created were, and format the dates in the UK standard format (e.g. 31/10/2008).
3. Write a query to find the WAREHOUSE_ID for all records in the D_WAREHOUSES table that were not created on 1/20/2009, for FCs outside of North America. (hint: you'll need to use TRUNC() and TO_DATE(). As of May 1, 2014, this returned 807 records.)
4. Edit the query to determine what day of the week each of those records were created.
5. Write a query to pull a list of all PO and ASINs, with their submitted quantities and order dates, for the vendor code 'DCCOM', during the date range of 1/1/2009 through 1/15/2009, in the US. Be sure to make use of partitioned columns in your WHERE clause, and run your query through Explain Plan before scheduling it.
6. Edit the query to sum up the quantity field by ASIN, removing the order date and PO fields.
7. Write a query against the table D_MP_ASINS_ESSENTIALS to pull the ASIN, ITEM_NAME, STREET_DAY, and PUBLICATION_DAY columns for any US Books ASIN with a PUBLICATION_DAY greater than or equal to 1/1/2020.
8. Edit the query to create an element that returns the STREET_DAY if it's not null, but returns the PUBLICATION_DAY if STREET_DAY is null. (hint: use the NVL() function to return pub date when street date is null.) This is a standard method used to determine release date.
9. Create a query that emails you a summary of all the POs you created the previous day, with count of ASINs and total units submitted for each PO, as well as any other details you're interested in, such as order type and vendor code (use hoot to find what fields are available). Schedule this query to run daily, and let it run for at least 7 days. (hint: you'll need to use D_DISTRIBUTOR_ORDER_ITEMS to get the ASIN level info.)

END