5

Managing Data in Different Time Zones

ORACLE

Objectives

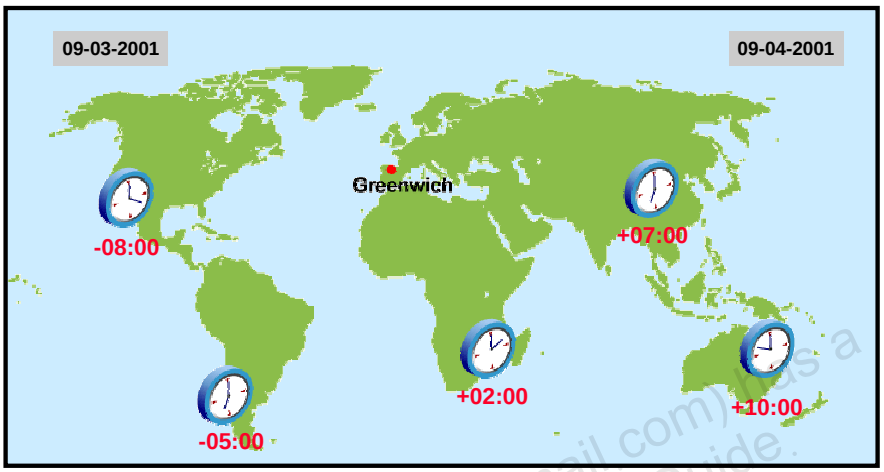
After completing this lesson, you should be able to use the following datetime functions:

- TZ_OFFSET
- FROM_TZ
- TO_TIMESTAMP
- TO_TIMESTAMP_TZ
- TO_YMINTERVAL
- TO_DSINTERVAL
- CURRENT_DATE
- CURRENT_TIMESTAMP
- LOCALTIMESTAMP
- DBTIMEZONE
- SESSIONTIMEZONE
- EXTRACT

Objectives

This lesson addresses some of the datetime functions available in the Oracle Database.

Time Zones



The image represents the time for each time zone when Greenwich time is 12:00.

ORACLE

Time Zones

The hours of the day are measured by the turning of the earth. The time of day at any particular moment depends on where you are. When it is noon in Greenwich, England, it is midnight along the International Date Line. The earth is divided into 24 time zones, one for each hour of the day. The time along the prime meridian in Greenwich, England, is known as Greenwich mean time, or GMT. GMT is the time standard against which all other time zones in the world are referenced. It is the same all year round and is not affected by summer time or daylight saving time. The meridian line is an imaginary line that runs from the North Pole to the South Pole. It is known as zero longitude and it is the line from which all other lines of longitude are measured. All time is measured relative to GMT and all places have a latitude (their distance north or south of the equator) and a longitude (their distance east or west of the Greenwich meridian).

TIME_ZONE Session Parameter

TIME_ZONE may be set to:

- An absolute offset
- Database time zone
- OS local time zone
- A named region

```
ALTER SESSION SET TIME_ZONE = '-05:00';
ALTER SESSION SET TIME_ZONE = dbtimezone;
ALTER SESSION SET TIME_ZONE = local;
ALTER SESSION SET TIME_ZONE = 'America/New_York';
```


ORACLE

TIME_ZONE Session Parameter

The Oracle Database supports storing the time zone in your date and time data, as well as fractional seconds. The ALTER SESSION command can be used to change time zone values in a user's session. The time zone values can be set to an absolute offset, a named time zone, a database time zone, or the local time zone.

CURRENT_DATE, CURRENT_TIMESTAMP, and LOCALTIMESTAMP

- CURRENT_DATE:
  - Returns the current date from the system
  - Has a data type of DATE
- CURRENT_TIMESTAMP:
  - Returns the current time stamp from the system
  - Has a data type of TIMESTAMP WITH TIME ZONE
- LOCALTIMESTAMP:
  - Returns the current time stamp from user session
  - Has a data type of TIMESTAMP

ORACLE

CURRENT_DATE, CURRENT_TIMESTAMP, and LOCALTIMESTAMP

The CURRENT_DATE and CURRENT_TIMESTAMP functions return the current date and current time stamp, respectively. The data type of CURRENT_DATE is DATE. The data type of CURRENT_TIMESTAMP is TIMESTAMP WITH TIME ZONE. The values returned display the time zone displacement of the SQL session executing the functions. The time zone displacement is the difference (in hours and minutes) between local time and Coordinated Universal Time (UTC). The TIMESTAMP WITH TIME ZONE data type has the format:

```
TIMESTAMP [ (fractional_seconds_precision) ] WITH TIME ZONE
```

where fractional_seconds_precision optionally specifies the number of digits in the fractional part of the SECOND datetime field and can be a number in the range 0 through 9. The default is 6.

The LOCALTIMESTAMP function returns the current date and time in the session time zone. The difference between LOCALTIMESTAMP and CURRENT_TIMESTAMP is that LOCALTIMESTAMP returns a TIMESTAMP value, whereas CURRENT_TIMESTAMP returns a TIMESTAMP WITH TIME ZONE value.

These functions are national language support (NLS) sensitive—that is, the results will be in the current NLS calendar and datetime formats.

CURRENT_DATE

Display the current date and time in the session's time zone.

```
ALTER SESSION
SET NLS_DATE_FORMAT = 'DD-MON-YYYY HH24:MI:SS';
```

```
ALTER SESSION SET TIME_ZONE = '-5:0';
SELECT SESSIONTIMEZONE, CURRENT_DATE FROM DUAL;
```

	SESSIONTIMEZONE	CURRENT_DATE
1	-05:00	18-NOV-2008 02:34:51

```
ALTER SESSION SET TIME_ZONE = '-8:0';
SELECT SESSIONTIMEZONE, CURRENT_DATE FROM DUAL;
```

	SESSIONTIMEZONE	CURRENT_DATE
1	-08:00	17-NOV-2008 23:37:03

ORACLE

CURRENT_DATE

The CURRENT_DATE function returns the current date in the session's time zone. The return value is a date in the Gregorian calendar.

The examples in the slide illustrate that CURRENT_DATE is sensitive to the session time zone. In the first example, the session is altered to set the TIME_ZONE parameter to –5:0. The TIME_ZONE parameter specifies the default local time zone displacement for the current SQL session. TIME_ZONE is a session parameter only, not an initialization parameter. The TIME_ZONE parameter is set as follows:

```
TIME_ZONE = '[+ | -] hh:mm'
```

The format mask ([+ | -] hh:mm) indicates the hours and minutes before or after UTC (Coordinated Universal Time, formerly known as Greenwich mean time).

Observe in the output that the value of CURRENT_DATE changes when the TIME_ZONE parameter value is changed to –8:0 in the second example.

Note: The ALTER SESSION command sets the date format of the session to 'DD-MON-YYYY HH24:MI:SS'—that is, day of month (1–31)-abbreviated name of month-4-digit year hour of day (0–23):minute (0–59):second (0–59).

CURRENT_TIMESTAMP

Display the current date and fractional time in the session's time zone.

```
ALTER SESSION SET TIME_ZONE = '-5:0';
SELECT SESSIONTIMEZONE, CURRENT_TIMESTAMP
FROM DUAL;
```

	SESSIONTIMEZONE	CURRENT_TIMESTAMP
1	-05:00	18-NOV-08 04.08.46.109757000 AM -05:00

```
ALTER SESSION SET TIME_ZONE = '-8:0';
SELECT SESSIONTIMEZONE, CURRENT_TIMESTAMP
FROM DUAL;
```

	SESSIONTIMEZONE	CURRENT_TIMESTAMP
1	-08:00	18-NOV-08 01.07.04.018151000 AM -08:00

ORACLE

CURRENT_TIMESTAMP

The CURRENT_TIMESTAMP function returns the current date and time in the session time zone, as a value of the TIMESTAMP WITH TIME ZONE data type. The time zone displacement reflects the current local time of the SQL session. The syntax of the CURRENT_TIMESTAMP function is:

```
CURRENT_TIMESTAMP (precision)
```

where *precision* is an optional argument that specifies the fractional second precision of the time value returned. If you omit precision, the default is 6.

The examples in the slide illustrate that CURRENT_TIMESTAMP is sensitive to the session time zone. In the first example, the session is altered to set the TIME_ZONE parameter to –5:0. Observe in the output that the value of CURRENT_TIMESTAMP changes when the TIME_ZONE parameter value is changed to –8:0 in the second example.

LOCALTIMESTAMP

- Display the current date and time in the session's time zone in a value of the TIMESTAMP data type.

```
ALTER SESSION SET TIME_ZONE = '-5:0';
SELECT CURRENT_TIMESTAMP, LOCALTIMESTAMP
FROM DUAL;
```

	CURRENT_TIMESTAMP	LOCALTIMESTAMP
1	18-NOV-08 04.10.19.678104000 AM -05:00	18-NOV-08 04.10.19.678104000 AM

```
ALTER SESSION SET TIME_ZONE = '-8:0';
SELECT CURRENT_TIMESTAMP, LOCALTIMESTAMP
FROM DUAL;
```

	CURRENT_TIMESTAMP	LOCALTIMESTAMP
1	18-NOV-08 01.11.33.170179000 AM -08:00	18-NOV-08 01.11.33.170179000 AM

- LOCALTIMESTAMP returns a TIMESTAMP value, whereas CURRENT_TIMESTAMP returns a TIMESTAMP WITH TIME ZONE value.

ORACLE

LOCALTIMESTAMP

The LOCALTIMESTAMP function returns the current date and time in the session time zone. LOCALTIMESTAMP returns a TIMESTAMP value. The syntax of the LOCALTIMESTAMP function is:

```
LOCALTIMESTAMP (TIMESTAMP_precision)
```

where TIMESTAMP precision is an optional argument that specifies the fractional second precision of the TIMESTAMP value returned.

The examples in the slide illustrate the difference between LOCALTIMESTAMP and CURRENT_TIMESTAMP. Observe that LOCALTIMESTAMP does not display the time zone value, whereas the CURRENT_TIMESTAMP does.

DBTIMEZONE and SESSIONTIMEZONE

- Display the value of the database time zone:

```
SELECT DBTIMEZONE FROM DUAL;
```

	DBTIMEZONE
1	+00:00

- Display the value of the session's time zone:

```
SELECT SESSIONTIMEZONE FROM DUAL;
```

	SESSIONTIMEZONE
1	-08:00

ORACLE

DBTIMEZONE and SESSIONTIMEZONE

The DBA sets the database's default time zone by specifying the SET TIME_ZONE clause of the CREATE DATABASE statement. If omitted, the default database time zone is the operating system time zone. The database time zone cannot be changed for a session with an ALTER SESSION statement.

The DBTIMEZONE function returns the value of the database time zone. The return type is a time zone offset (a character type in the format: '[+|-]TZH:TZM') or a time zone region name, depending on how the user specified the database time zone value in the most recent CREATE DATABASE or ALTER DATABASE statement. The example in the slide shows that the database time zone is set to "–05:00," as the TIME_ZONE parameter is in the format:

```
TIME_ZONE = '[+ | -] hh:mm'
```

The SESSIONTIMEZONE function returns the value of the current session's time zone. The return type is a time zone offset (a character type in the format '[+|-]TZH:TZM') or a time zone region name, depending on how the user specified the session time zone value in the most recent ALTER SESSION statement. The example in the slide shows that the session time zone is offset to UTC by –8 hours. Observe that the database time zone is different from the current session's time zone.

TIMESTAMP Data Type

- The TIMESTAMP data type is an extension of the DATE data type.
- It stores the year, month, and day of the DATE data type, plus hour, minute, and second values, as well as the fractional second value.
- Variations in TIMESTAMP are:
  - `TIMESTAMP [(fractional_seconds_precision)]_`
  - `TIMESTAMP [(fractional_seconds_precision)]_ WITH TIME ZONE`
  - `TIMESTAMP [(fractional_seconds_precision)]_ WITH LOCAL TIME ZONE`

Datetime Data Types

The TIMESTAMP data type contains the datetime fields: YEAR, MONTH, DAY, HOUR, MINUTE, and SECOND and fractional seconds.

The TIMESTAMP WITH TIME ZONE data type contains the datetime fields: YEAR, MONTH, DAY, HOUR, MINUTE, SECOND, TIMEZONE_HOUR, and TIMEZONE_MINUTE and fractional seconds.

The TIMESTAMP WITH LOCAL TIME ZONE data type contains the same information as the TIMESTAMP data type, except that the data is normalized to the database time zone when stored, and adjusted to match the client's time zone when retrieved.

Note: Fractional second precision specifies the number of digits in the fractional part of the SECOND datetime field and can be a number in the range 0 through 9. The default is 6.

TIMESTAMP Data Types

Data Type	Fields
TIMESTAMP	Year, Month, Day, Hour, Minute, Second with fractional seconds
TIMESTAMP WITH TIME ZONE	Same as the TIMESTAMP data type; also includes: TIMEZONE_HOUR, and TIMEZONE_MINUTE or TIMEZONE_REGION
TIMESTAMP WITH LOCAL TIME ZONE	Same as the TIMESTAMP data type; also includes a time zone offset in its value

TIMESTAMP Data Types

TIMESTAMP (fractional_seconds_ precision)

This data type contains the year, month, and day values of date, as well as hour, minute, and second values of time, where significant fractional seconds precision is the number of digits in the fractional part of the SECOND datetime field. The accepted values of significant fractional_seconds_precision are 0 through 9. The default is 6.

TIMESTAMP (fractional_seconds_precision) WITH TIME ZONE

This data type contains all values of TIMESTAMP as well as time zone displacement value.

TIMESTAMP (fractional_seconds_precision) WITH LOCAL TIME ZONE

This data type contains all values of TIMESTAMP, with the following exceptions:

- Data is normalized to the database time zone when it is stored in the database.
- When the data is retrieved, users see the data in the session time zone.

TIMESTAMP Fields

Datetime Field	Valid Values
YEAR	–4712 to 9999 (excluding year 0)
MONTH	01 to 12
DAY	01 to 31
HOUR	00 to 23
MINUTE	00 to 59
SECOND	00 to 59.9(N) where 9(N) is precision
TIMEZONE_HOUR	–12 to 14
TIMEZONE_MINUTE	00 to 59

ORACLE

TIMESTAMP Fields

Each datetime data type is composed of several of these fields. Datetimes are mutually comparable and assignable only if they have the same datetime fields.

Difference Between DATE and TIMESTAMP

A

```
-- when hire_date is
of type DATE

SELECT hire_date
FROM emp5;
```

	HIRE_DATE
1	17-JUN-1987 00:00:00
2	21-SEP-1989 00:00:00
3	13-JAN-1993 00:00:00
4	03-JAN-1990 00:00:00
5	21-MAY-1991 00:00:00
6	25-JUN-1997 00:00:00

...

B

```
ALTER TABLE emp5
MODIFY hire_date TIMESTAMP;

SELECT hire_date
FROM emp5;
```

	HIRE_DATE
1	17-JUN-87 12.00.00.000000000 AM
2	21-SEP-89 12.00.00.000000000 AM
3	13-JAN-93 12.00.00.000000000 AM
4	03-JAN-90 12.00.00.000000000 AM
5	21-MAY-91 12.00.00.000000000 AM
6	25-JUN-97 12.00.00.000000000 AM

...

TIMESTAMP Data Type: Example

In the slide, example A shows the data from the `hire_date` column of the `EMP5` table when the data type of the column is `DATE`. In example B, the table is altered and the data type of the `hire_date` column is made into `TIMESTAMP`. The output shows the differences in display. You can convert from `DATE` to `TIMESTAMP` when the column has data, but you cannot convert from `DATE` or `TIMESTAMP` to `TIMESTAMP WITH TIME ZONE` unless the column is empty.

You can specify the fractional seconds precision for time stamp. If none is specified, as in this example, then it defaults to 6.

For example, the following statement sets the fractional seconds precision as 7:

```
ALTER TABLE emp5
MODIFY hire_date TIMESTAMP(7);
```

Note: The Oracle date data type by default appears as shown in this example. However, the date data type also contains additional information such as hours, minutes, seconds, a.m., and p.m. To obtain the date in this format, you can apply a format mask or a function to the date value.

TIMESTAMP WITH TIME ZONE Data Type

- TIMESTAMP WITH TIME ZONE is a variant of TIMESTAMP that includes a time zone displacement in its value.
- The time zone displacement is the difference, in hours and minutes, between local time and UTC.
- It is specified as:

```
TIMESTAMP[(fractional_seconds_precision)]
WITH TIME ZONE
```


ORACLE

TIMESTAMP WITH TIME ZONE Data Type

UTC stands for Coordinated Universal Time (formerly Greenwich mean time). Two TIMESTAMP WITH TIME ZONE values are considered identical if they represent the same instant in UTC, regardless of the TIME ZONE offsets stored in the data. For example:

```
TIMESTAMP '1999-04-15 8:00:00 -8:00'
```

is the same as

```
TIMESTAMP '1999-04-15 11:00:00 -5:00'.
```

That is, 8:00 a.m. Pacific Standard Time is the same as 11:00 a.m. Eastern Standard Time.

This can also be specified as:

```
TIMESTAMP '1999-04-15 8:00:00 US/Pacific'
```

TIMESTAMP WITH TIMEZONE: Example

```
CREATE TABLE web_orders
(ord_id number primary key,
 order_date TIMESTAMP WITH TIME ZONE);
```

```
INSERT INTO web_orders values
(ord_seq.nextval, current_date);
```

```
SELECT * FROM web_orders;
```

	ORD_ID	ORDER_DATE
1	100	18-NOV-08 01.27.52.000000000 AM -08:00

ORACLE

TIMESTAMP WITH TIME ZONE: Example

In the example in the slide, a new table `web_orders` is created with a column of data type `TIMESTAMP WITH TIME ZONE`. This table is populated whenever a `web_order` is placed. The time stamp and time zone for the user placing the order is inserted based on the `CURRENT_DATE value`. That way when a Web-based company guarantees shipping, they can estimate their delivery time based on the time zone of the person placing the order.

TIMESTAMP WITH LOCAL TIMEZONE

- TIMESTAMP WITH LOCAL TIME ZONE is another variant of TIMESTAMP that includes a time zone displacement in its value.
- Data stored in the database is normalized to the database time zone.
- The time zone displacement is not stored as part of the column data.
- The Oracle Database returns the data in the user's local session time zone.
- The TIMESTAMP WITH LOCAL TIME ZONE data type is specified as follows:

```
TIMESTAMP[(fractional_seconds_precision)]
WITH LOCAL TIME ZONE
```

ORACLE

TIMESTAMP WITH LOCAL TIMEZONE

Unlike TIMESTAMP WITH TIME ZONE, you can specify columns of the TIMESTAMP WITH LOCAL TIME ZONE type as part of a primary or unique key. The time zone displacement is the difference (in hours and minutes) between local time and UTC. There is no literal for TIMESTAMP WITH LOCAL TIME ZONE.

TIMESTAMP WITH LOCAL TIMEZONE: Example

```
CREATE TABLE shipping (delivery_time TIMESTAMP WITH
LOCAL TIME ZONE);
INSERT INTO shipping VALUES(current_timestamp + 2);
```

```
SELECT * FROM shipping;
```

	DELIVERY_TIME
1	20-NOV-08 01.30.15.000000000 AM

```
ALTER SESSION SET TIME_ZONE = 'EUROPE/LONDON';
SELECT * FROM shipping;
```

	DELIVERY_TIME
1	20-NOV-08 09.30.15.000000000 AM

TIMESTAMP WITH LOCAL TIME ZONE: Example

In the example in the slide, a new table SHIPPING is created with a column of the TIMESTAMP WITH LOCAL TIME ZONE data type. This table is populated by inserting two days from the CURRENT_TIMESTAMP value into it every time an order is placed. The output from the DATE_TAB table shows that the data is stored without the time zone offset. Then the ALTER SESSION command is issued to change the time zone to the local time zone at the place of delivery. A second query on the same table now reflects the data with the local time zone reflected in the time value, so that the customer can be notified about the expected delivery time.

INTERVAL Data Types

- INTERVAL data types are used to store the difference between two datetime values.
- There are two classes of intervals:
  - Year-month
  - Day-time
- The precision of the interval is:
  - The actual subset of fields that constitutes an interval
  - Specified in the interval qualifier

Data Type	Fields
INTERVAL YEAR TO MONTH	Year, Month
INTERVAL DAY TO SECOND	Days, Hour, Minute, Second with fractional seconds

ORACLE

INTERVAL Data Types

INTERVAL data types are used to store the difference between two datetime values. There are two classes of intervals: year-month intervals and day-time intervals. A year-month interval is made up of a contiguous subset of fields of YEAR and MONTH, whereas a day-time interval is made up of a contiguous subset of fields consisting of DAY, HOUR, MINUTE, and SECOND. The actual subset of fields that constitute an interval is called the precision of the interval and is specified in the interval qualifier. Because the number of days in a year is calendar dependent, the year-month interval is NLS dependent, whereas day-time interval is NLS independent.

The interval qualifier may also specify the leading field precision, which is the number of digits in the leading or only field, and in case the trailing field is SECOND, it may also specify the fractional seconds precision, which is the number of digits in the fractional part of the SECOND value. If not specified, the default value for leading field precision is 2 digits, and the default value for fractional seconds precision is 6 digits.

INTERVAL Data Types (continued)

INTERVAL YEAR (year_precision) TO MONTH

This data type stores a period of time in years and months, where `year_precision` is the number of digits in the YEAR datetime field. The accepted values are 0 to 9. The default is 6.

INTERVAL DAY (day_precision) TO SECOND (fractional_seconds_precision)

This data type stores a period of time in days, hours, minutes, and seconds, where `day_precision` is the maximum number of digits in the DAY datetime field (accepted values are 0 to 9; the default is 2), and `fractional_seconds_precision` is the number of digits in the fractional part of the SECOND field. The accepted values are 0 to 9. The default is 6.

INTERVAL Fields

INTERVAL Field	Valid Values for Interval
YEAR	Any positive or negative integer
MONTH	00 to 11
DAY	Any positive or negative integer
HOUR	00 to 23
MINUTE	00 to 59
SECOND	00 to 59.9(N) where 9(N) is precision

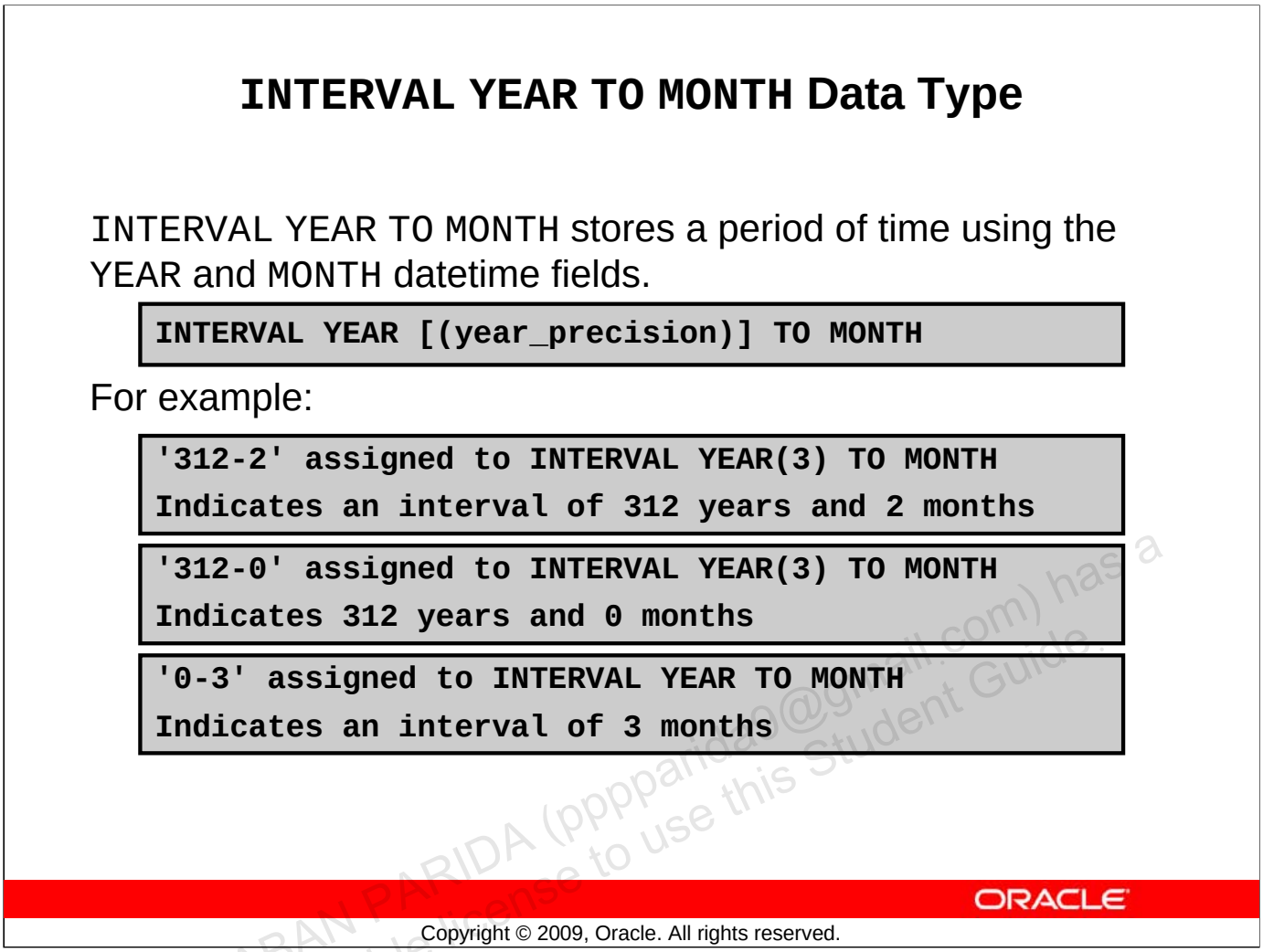
INTERVAL Fields

INTERVAL YEAR TO MONTH can have fields of YEAR and MONTH.

INTERVAL DAY TO SECOND can have fields of DAY, HOUR, MINUTE, and SECOND.

The actual subset of fields that constitute an item of either type of interval is defined by an interval qualifier, and this subset is known as the precision of the item.

Year-month intervals are mutually comparable and assignable only with other year-month intervals, and day-time intervals are mutually comparable and assignable only with other day-time intervals.



INTERVAL YEAR TO MONTH Data Type

INTERVAL YEAR TO MONTH stores a period of time using the YEAR and MONTH datetime fields. Specify INTERVAL YEAR TO MONTH as follows:

```
INTERVAL YEAR [(year_precision)] TO MONTH
```

where year_precision is the number of digits in the YEAR datetime field. The default value of year_precision is 2.

Restriction: The leading field must be more significant than the trailing field. For example, INTERVAL '0-1' MONTH TO YEAR is not valid.

The following INTERVAL YEAR TO MONTH literal indicates an interval of 123 years, 3 months:

- INTERVAL '123-3' YEAR(3) TO MONTH
- INTERVAL '123' YEAR(3) indicates an interval of 123 years 0 months.
- INTERVAL '3' MONTH indicates an interval of 3 months.

INTERVAL YEAR TO MONTH: Example

```
CREATE TABLE warranty
(prod_id number,  warranty_time INTERVAL YEAR(3)
TO MONTH);
INSERT INTO warranty VALUES (123, INTERVAL '8'
MONTH);
INSERT INTO warranty VALUES (155, INTERVAL '200'
YEAR(3));
INSERT INTO warranty VALUES (678, '200-11');
SELECT * FROM warranty;
```

	PROD_ID	WARRANTY_TIME
1	123	0-8
2	155	200-0
3	678	200-11

ORACLE

INTERVAL YEAR TO MONTH Data Type (continued)

INTERVAL YEAR TO MONTH stores a period of time using the YEAR and MONTH datetime fields. Specify INTERVAL YEAR TO MONTH as follows:

```
INTERVAL YEAR [(year_precision)] TO MONTH
```

where year_precision is the number of digits in the YEAR datetime field. The default value of year_precision is 2.

Restriction: The leading field must be more significant than the trailing field. For example, INTERVAL '0-1' MONTH TO YEAR is not valid.

The Oracle Database supports two interval data types: Interval Year to Month and Interval Day to Second; the column type, PL/SQL argument, variable, and return type must be one of the two. However, for interval literals, the system recognizes other American National Standards Institute (ANSI) interval types such as INTERVAL '2' YEAR or INTERVAL '10' HOUR. In these cases, each interval is converted to one of the two supported types.

In the example in the slide, a WARRANTY table is created, which contains a warranty_time column that takes the INTERVAL YEAR(3) TO MONTH data type. Different values are inserted into it to indicate years and months for various products. When these rows are retrieved from the table, you see a year value displaced by the month value by a (-).

INTERVAL DAY TO SECOND Data Type

INTERVAL DAY TO SECOND (fractional_seconds_precision)stores a period of time in days, hours, minutes, and seconds.

```
INTERVAL DAY[(day_precision)] TO Second
```

For example:

```
INTERVAL '6 03:30:16' DAY TO SECOND
Indicates an interval of 6 days 3 hours 30 minutes
and 16 seconds
```

```
INTERVAL '6 00:00:00' DAY TO SECOND
Indicates an interval of 6 days and 0 hours,  0
minutes and 0 seconds
```


INTERVAL DAY TO SECOND Data Type

INTERVAL DAY (day_precision) TO SECOND(fractional_seconds_precision) stores a period of time in days, hours, minutes, and seconds, where day_precision is the maximum number of digits in the DAY datetime field (accepted values are 0 through 9; the default is 2), and fractional_seconds_precision is the number of digits in the fractional part of the SECOND field. Accepted values are 0 through 9. The default is 6.

In the example in the slide, 6 represents the number of days, and 03:30:15 indicates the values for hours, minutes, and seconds.

INTERVAL DAY TO SECOND Data Type: Example

```
CREATE TABLE lab
( exp_id number, test_time INTERVAL DAY(2) TO
SECOND);

INSERT INTO lab VALUES (100012, '90 00:00:00');
INSERT INTO lab VALUES (56098,
  INTERVAL '6 03:30:16' DAY  TO SECOND);
```

```
SELECT * FROM lab;
```

	EXP_ID	TEST_TIME
1	100012	90 0:0:0.0
2	56098	6 3:30:16.0

ORACLE

INTERVAL DAY TO SECOND Data Type: Example

In the example in the slide, you are creating the lab table with a `test_time` column of the `INTERVAL DAY TO SECOND` data type. You then insert into it the value “90 00:00:00” to indicate 90 days and 0 hours, minutes, and seconds, and `INTERVAL '6 03:30:16' DAY TO SECOND`. The select statement shows how this data is displayed in the database.

EXTRACT

- Display the YEAR component from the SYSDATE.

```
SELECT EXTRACT (YEAR FROM SYSDATE) FROM DUAL;
```

	EXTRACT(YEARFROMSYSDATE)
1	2008

- Display the MONTH component from the HIRE_DATE for those employees whose MANAGER_ID is 100.

```
SELECT last_name, hire_date,
       EXTRACT (MONTH FROM HIRE_DATE)
FROM employees
WHERE manager_id = 100;
```

	LAST_NAME	HIRE_DATE	EXTRACT(MONTHFROMHIRE_DATE)
1	Kochhar	21-SEP-1989 00:00:00	9
2	De Haan	13-JAN-1993 00:00:00	1
3	Raphaely	07-DEC-1994 00:00:00	12
4	Weiss	18-JUL-1996 00:00:00	7

...

ORACLE

EXTRACT

The EXTRACT expression extracts and returns the value of a specified datetime field from a datetime or interval value expression. You can extract any of the components mentioned in the following syntax using the EXTRACT function. The syntax of the EXTRACT function is:

```
SELECT  EXTRACT ([YEAR] [MONTH][DAY] [HOUR] [MINUTE][SECOND]
                 [TIMEZONE_HOUR] [TIMEZONE_MINUTE]
                 [TIMEZONE_REGION] [TIMEZONE_ABBR]
FROM [datetime_value_expression] [interval_value_expression]);
```

When you extract a TIMEZONE_REGION or TIMEZONE_ABBR (abbreviation), the value returned is a string containing the appropriate time zone name or abbreviation. When you extract any of the other values, the value returned is a date in the Gregorian calendar. When extracting from a datetime with a time zone value, the value returned is in UTC.

In the first example in the slide, the EXTRACT function is used to extract the YEAR from SYSDATE. In the second example in the slide, the EXTRACT function is used to extract the MONTH from the HIRE_DATE column of the EMPLOYEES table for those employees who report to the manager whose EMPLOYEE_ID is 100.

TZ_OFFSET

- Display the time zone offset for the `'US/Eastern'` time zone:

```
SELECT TZ_OFFSET('US/Eastern') FROM DUAL;
```

	TZ_OFFSET('US/EASTERN')
1	-05:00

- Display the time zone offset for the `'Canada/Yukon'` time zone:

```
SELECT TZ_OFFSET('Canada/Yukon') FROM DUAL;
```

	TZ_OFFSET('CANADA/YUKON')
1	-08:00

- Display the time zone offset for the `'Europe/London'` time zone:

```
SELECT TZ_OFFSET('Europe/London') FROM DUAL;
```

	TZ_OFFSET('EUROPE/LONDON')
1	+00:00

ORACLE

TZ_OFFSET

The `TZ_OFFSET` function returns the time zone offset corresponding to the value entered. The return value is dependent on the date when the statement is executed. For example, if the `TZ_OFFSET` function returns a value –08:00, this value indicates that the time zone where the command was executed is eight hours behind UTC. You can enter a valid time zone name, a time zone offset from UTC (which simply returns itself), or the keyword `SESSIONTIMEZONE` or `DBTIMEZONE`. The syntax of the `TZ_OFFSET` function is:

```
TZ_OFFSET ( ['time_zone_name'] '[+ | -] hh:mm']
            [ SESSIONTIMEZONE] [DBTIMEZONE]
```

The Fold Motor Company has its headquarters in Michigan, USA, which is in the US/Eastern time zone. The company president, Mr. Fold, wants to conduct a conference call with the vice president of the Canadian operations and the vice president of European operations, who are in the Canada/Yukon and Europe/London time zones, respectively. Mr. Fold wants to find out the time in each of these places to make sure that his senior management will be available to attend the meeting. His secretary, Mr. Scott, helps by issuing the queries shown in the example and gets the following results:

- The `'US/Eastern'` time zone is five hours behind UTC.
- The `'Canada/Yukon'` time zone is eight hours behind UTC.
- The `'Europe/London'` time zone is same as UTC.

TZ_OFFSET (continued)

For a listing of valid time zone name values, you can query the V$TIMEZONE_NAMES dynamic performance view.

```
SELECT * FROM V$TIMEZONE_NAMES;
```

	TZNAME	TZABBREV
1	Africa/Algiers	LMT
2	Africa/Algiers	PMT
3	Africa/Algiers	WET
4	Africa/Algiers	WEST
5	Africa/Algiers	CET
6	Africa/Algiers	CEST
7	Africa/Cairo	LMT
8	Africa/Cairo	EET
9	Africa/Cairo	EEST
10	Africa/Casablanca	LMT
11	Africa/Casablanca	WET
12	Africa/Casablanca	WEST
13	Africa/Casablanca	CET
14	Africa/Ceuta	LMT
15	Africa/Ceuta	WET

...

TIMESTAMP Conversion Using FROM_TZ

- Display the TIMESTAMP value '2000-03-28 08:00:00' as a TIMESTAMP WITH TIME ZONE value.

```
SELECT FROM_TZ(TIMESTAMP
       '2000-03-28 08:00:00','3:00')
FROM DUAL;
```

	FROM_TZ(TIMESTAMP'2000-03-2808:00:00','3:00')
1	28-MAR-00 08.00.00.000000000 AM +03:00

- Display the TIMESTAMP value '2000-03-28 08:00:00' as a TIMESTAMP WITH TIME ZONE value for the 'Australia/North' time zone region.

```
SELECT FROM_TZ(TIMESTAMP
       '2000-03-28 08:00:00', 'Australia/North')
FROM DUAL;
```

	FROM_TZ(TIMESTAMP'2000-03-2808:00:00','AUSTRALIA/NORTH')
1	28-MAR-00 08.00.00.000000000 AM AUSTRALIA/NORTH

ORACLE

TIMESTAMP Conversion Using FROM_TZ

The FROM_TZ function converts a TIMESTAMP value to a TIMESTAMP WITH TIME ZONE value.

The syntax of the FROM_TZ function is as follows:

```
FROM_TZ(TIMESTAMP timestamp_value, time_zone_value)
```

where time_zone_value is a character string in the format 'TZH:TZM' or a character expression that returns a string in TZR (time zone region) with an optional TZD format. TZD is an abbreviated time zone string with daylight saving information. TZR represents the time zone region in datetime input strings. Examples are 'Australia/North', 'PST' for US/Pacific standard time, 'PDT' for US/Pacific daylight time, and so on. To see a listing of valid values for the TZR and TZD format elements, query the V$TIMEZONE_NAMES dynamic performance view.

The example in the slide converts a TIMESTAMP value to TIMESTAMP WITH TIME ZONE.

Converting to TIMESTAMP Using TO_TIMESTAMP and TO_TIMESTAMP_TZ

- Display the character string '2000-12-01 11:00:00' as a TIMESTAMP value:

```
SELECT TO_TIMESTAMP ('2000-12-01 11:00:00',
                     'YYYY-MM-DD HH:MI:SS')
FROM DUAL;
```

	TO_TIMESTAMP('2000-12-0111:00:00','YYYY-MM-DDHH:MI:SS')
1	01-DEC-00 11.00.00.000000000 AM

- Display the character string '1999-12-01 11:00:00 -8:00' as a TIMESTAMP WITH TIME ZONE value:

```
SELECT
  TO_TIMESTAMP_TZ('1999-12-01 11:00:00 -8:00',
                  'YYYY-MM-DD HH:MI:SS TZH:TZM')
FROM DUAL;
```

	TO_TIMESTAMP_TZ('1999-12-0111:00:00-8:00','YYYY-MM-DDHH:MI:SSTZH:TZM')
1	01-DEC-99 11.00.00.000000000 AM -08:00

ORACLE

Converting to TIMESTAMP Using TO_TIMESTAMP and TO_TIMESTAMP_TZ

The TO_TIMESTAMP function converts a string of CHAR, VARCHAR2, NCHAR, or NVARCHAR2 data type to a value of the TIMESTAMP data type. The syntax of the TO_TIMESTAMP function is:

```
TO_TIMESTAMP (char,[fmt],['nlsparam'])
```

The optional fmt specifies the format of char. If you omit fmt, then the string must be in the default format of the TIMESTAMP data type. The optional nlsparam specifies the language in which month and day names, and abbreviations are returned. This argument can have this form:

```
'NLS_DATE_LANGUAGE = language'
```

If you omit nlsparams, this function uses the default date language for your session. The example in the slide converts a character string to a value of TIMESTAMP.

The TO_TIMESTAMP_TZ function converts a string of CHAR, VARCHAR2, NCHAR, or NVARCHAR2 data type to a value of the TIMESTAMP WITH TIME ZONE data type. The syntax of the TO_TIMESTAMP_TZ function is:

```
TO_TIMESTAMP_TZ (char,[fmt],['nlsparam'])
```

The optional fmt specifies the format of char. If you omit fmt, then a string must be in the default format of the TIMESTAMP WITH TIME ZONE data type. The example in the slide converts a character string to a value of TIMESTAMP WITH TIME ZONE.

Time Interval Conversion with TO_YMINTERVAL

Display a date that is one year and two months after the hire date for the employees working in the department with the DEPARTMENT_ID 20.

```
SELECT hire_date,
       hire_date + TO_YMINTERVAL('01-02') AS
       HIRE_DATE_YMININTERVAL
FROM   employees
WHERE department_id = 20;
```

	HIRE_DATE	HIRE_DATE_YMININTERVAL
1	17-FEB-1996 00:00:00	17-APR-1997 00:00:00
2	17-AUG-1997 00:00:00	17-OCT-1998 00:00:00

ORACLE

Time Interval Conversion with TO_YMINTERVAL

The TO_YMINTERVAL function converts a character string of CHAR, VARCHAR2, NCHAR, or NVARCHAR2 data type to an INTERVAL YEAR TO MONTH data type. The INTERVAL YEAR TO MONTH data type stores a period of time using the YEAR and MONTH datetime fields. The format of INTERVAL YEAR TO MONTH is as follows:

```
INTERVAL YEAR [(year_precision)] TO MONTH
```

where year_precision is the number of digits in the YEAR datetime field. The default value of year_precision is 2.

The syntax of the TO_YMINTERVAL function is:

```
TO_YMINTERVAL (char)
```

where char is the character string to be converted.

The example in the slide calculates a date that is one year and two months after the hire date for the employees working in the department 20 of the EMPLOYEES table.

A reverse calculation can also be done using the TO_YMINTERVAL function. For example:

```
SELECT hire_date, hire_date + TO_YMINTERVAL('-02-04') AS
       HIRE_DATE_YMINTERVAL
FROM EMPLOYEES WHERE department_id = 20;
```

Observe that the character string passed to the TO_YMINTERVAL function has a negative value. The example returns a date that is two years and four months before the hire date for the employees working in the department 20 of the EMPLOYEES table.

Using TO_DSINTERVAL: Example

TO_DSINTERVAL: Converts a character string to an INTERVAL DAY TO SECOND data type

```
SELECT last_name,
 TO_CHAR(hire_date, 'mm-dd-yy:hh:mi:ss') hire_date,
  TO_CHAR(hire_date +
   TO_DSINTERVAL('100 10:00:00'),
     'mm-dd-yy:hh:mi:ss') hiredate2
FROM employees;
```

	LAST_NAME	HIRE_DATE	HIREDATE2
1	King	06-17-87:12:00:00	09-25-87:10:00:00
2	Kochhar	09-21-89:12:00:00	12-30-89:10:00:00
3	De Haan	01-13-93:12:00:00	04-23-93:10:00:00
4	Hunold	01-03-90:12:00:00	04-13-90:10:00:00
5	Ernst	05-21-91:12:00:00	08-29-91:10:00:00
6	Austin	06-25-97:12:00:00	10-03-97:10:00:00
7	Pataballa	02-05-98:12:00:00	05-16-98:10:00:00
8	Lorentz	02-07-99:12:00:00	05-18-99:10:00:00

...

ORACLE

TO_DSINTERVAL

TO_DSINTERVAL converts a character string of the CHAR, VARCHAR2, NCHAR, or NVARCHAR2 data type to an INTERVAL DAY TO SECOND type.

In the example in the slide, the date 100 days and 10 hours after the hire date is obtained.

TO_YMINTERVAL

The TO_YMINTERVAL function converts a character string of CHAR, VARCHAR2, NCHAR, or NVARCHAR2 data type to an INTERVAL YEAR TO MONTH type.

In the following example, the date one year and two months after the hire date is obtained.

```
SELECT hire_date, hire_date + TO_YMINTERVAL('01-02') ytm
FROM employees;
```

	HIRE_DATE	YTM
1	17-JUN-1987 00:00:00	17-AUG-1988 00:00:00
2	21-SEP-1989 00:00:00	21-NOV-1990 00:00:00
3	13-JAN-1993 00:00:00	13-MAR-1994 00:00:00
4	03-JAN-1990 00:00:00	03-MAR-1991 00:00:00
5	21-MAY-1991 00:00:00	21-JUL-1992 00:00:00

...

Daylight Saving Time

- First Sunday in April
  - Time jumps from 01:59:59 a.m. to 03:00:00 a.m.
  - Values from 02:00:00 a.m. to 02:59:59 a.m. are not valid.
- Last Sunday in October
  - Time jumps from 02:00:00 a.m. to 01:00:01 a.m.
  - Values from 01:00:01 a.m. to 02:00:00 a.m. are ambiguous because they are visited twice.

ORACLE

Daylight Saving Time (DST)

Most western nations advance the clock ahead one hour during the summer months. This period is called daylight saving time. Daylight saving time lasts from the first Sunday in April to the last Sunday in October in the most of the United States, Mexico, and Canada. The nations of the European Union observe daylight saving time, but they call it the summer time period. Europe's summer time period begins a week earlier than its North American counterpart, but ends at the same time.

The Oracle Database automatically determines, for any given time zone region, whether daylight saving time is in effect and returns local time values accordingly. The datetime value is sufficient for the Oracle Database to determine whether daylight saving time is in effect for a given region in all cases except boundary cases. A boundary case occurs during the period when daylight saving time goes into or out of effect. For example, in the US/Eastern region, when daylight saving time goes into effect, the time changes from 01:59:59 a.m. to 03:00:00 a.m. The one-hour interval between 02:00:00 and 02:59:59 a.m. does not exist. When daylight saving time goes out of effect, the time changes from 02:00:00 a.m. back to 01:00:01 a.m., and the one-hour interval between 01:00:01 and 02:00:00 a.m. is repeated.

Daylight Saving Time (DST) (continued)

ERROR_ON_OVERLAP_TIME

The ERROR_ON_OVERLAP_TIME is a session parameter to notify the system to issue an error when it encounters a datetime that occurs in the overlapped period and no time zone abbreviation was specified to distinguish the period.

For example, daylight saving time ends on October 31, at 02:00:01 a.m. The overlapped periods are:

- 10/31/2004 01:00:01 a.m. to 10/31/2004 02:00:00 a.m. (EDT)
- 10/31/2004 01:00:01 a.m. to 10/31/2004 02:00:00 a.m. (EST)

If you input a datetime string that occurs in one of these two periods, you need to specify the time zone abbreviation (for example, EDT or EST) in the input string for the system to determine the period. Without this time zone abbreviation, the system does the following:

If the parameter ERROR_ON_OVERLAP_TIME is FALSE, then it assumes that the input time is standard time (for example, EST). Otherwise, an error is raised.

Summary

In this lesson, you should have learned how to use the following functions:

- TZ_OFFSET
- FROM_TZ
- TO_TIMESTAMP
- TO_TIMESTAMP_TZ
- TO_YMINTERVAL
- CURRENT_DATE
- CURRENT_TIMESTAMP
- LOCALTIMESTAMP
- DBTIMEZONE
- SESSIONTIMEZONE
- EXTRACT

Summary

This lesson addressed some of the datetime functions available in the Oracle Database.

Practice 5: Overview

This practice covers using the datetime functions.

Practice 5: Overview

In this practice, you display time zone offsets, `CURRENT_DATE`, `CURRENT_TIMESTAMP`, and `LOCALTIMESTAMP`. You also set time zones and use the `EXTRACT` function.

Practice 5

1. Alter the session to set the `NLS_DATE_FORMAT` to `DD-MON-YYYY HH24:MI:SS`.
2. a. Write queries to display the time zone offsets (`TZ_OFFSET`) for the following time zones.
   - *US/Pacific-New*

	TZ_OFFSET('US/PACIFIC-NEW')
1	-08:00

   - *Singapore*

	TZ_OFFSET('SINGAPORE')
1	+08:00

   - *Egypt*

	TZ_OFFSET('EGYPT')
1	+02:00

   b. Alter the session to set the `TIME_ZONE` parameter value to the time zone offset of US/Pacific-New.
   c. Display the `CURRENT_DATE`, `CURRENT_TIMESTAMP`, and `LOCALTIMESTAMP` for this session.

	CURRENT_DATE	CURRENT_TIMESTAMP	LOCALTIMESTAMP
1	18-NOV-2008 03:17:31	18-NOV-08 03.17.31.389656000 AM -07:00	18-NOV-08 03.17.31.389656000 AM

   d. Alter the session to set the `TIME_ZONE` parameter value to the time zone offset of Singapore.
   e. Display the `CURRENT_DATE`, `CURRENT_TIMESTAMP`, and `LOCALTIMESTAMP` for this session.
   Note: The output may be different based on the date when the command is executed.

	CURRENT_DATE	CURRENT_TIMESTAMP	LOCALTIMESTAMP
1	18-NOV-2008 18:19:42	18-NOV-08 06.19.42.486210000 PM +08:00	18-NOV-08 06.19.42.486210000 PM

   Note: Observe in the preceding practice that `CURRENT_DATE`, `CURRENT_TIMESTAMP`, and `LOCALTIMESTAMP` are all sensitive to the session time zone.

3. Write a query to display `DBTIMEZONE` and `SESSIONTIMEZONE`.

	DBTIMEZONE	SESSIONTIMEZONE
1	+00:00	+08:00

Practice 5 (continued)

4. Write a query to extract YEAR from the HIRE_DATE column of the EMPLOYEES table for those employees who work in department 80.

	LAST_NAME	EXTRACT(YEARFROMHIRE_DATE)
1	Russell	1996
2	Partners	1997
3	Errazuriz	1997
4	Cambrault	1999
5	Zlotkey	2000
6	Tucker	1997
7	Bernstein	1997

▪▪▪

30	Abel	1996
31	Hutton	1997
32	Taylor	1998
33	Livingston	1998
34	Johnson	2000

5. Alter the session to set NLS_DATE_FORMAT to DD-MON-YYYY.
6. Examine and run the lab_05_06.sql script to create the SAMPLE_DATES table and populate it.
   a. Select from the table and view the data.

	DATE_COL
1	18-NOV-2008

   b. Change the data type of the DATE_COL column to TIMESTAMP. Select from the table to view the data.

	DATE_COL
1	18-NOV-08 05.31.54.000000000 AM

   c. Try to change the data type of the DATE_COL column to TIMESTAMP WITH TIME ZONE. What happens?

Practice 5 (continued)

7. Create a query to retrieve last names from the `EMPLOYEES` table and calculate the review status. If the year hired is 1998, display `Needs Review` for the review status; otherwise, display `not this year!` Name the review status column `Review`. Sort the results by the `HIRE_DATE` column.
   Hint: Use a `CASE` expression with the `EXTRACT` function to calculate the review status.

	LAST_NAME	Review
1	King	not this year!
2	Whalen	not this year!
3	Kochhar	not this year!
4	Hunold	not this year!
5	Ernst	not this year!
6	De Haan	not this year!
7	Mavris	not this year!
8	Baer	not this year!

▪▪▪

65	Olsen	Needs Review
66	Patel	Needs Review
67	Livingston	Needs Review
68	Walsh	Needs Review
69	Feeney	Needs Review
70	Dellinger	Needs Review
71	McCain	Needs Review
72	Vargas	Needs Review
73	Gates	Needs Review
74	Rogers	Needs Review

▪▪▪

105	Ande	not this year!
106	Banda	not this year!
107	Kumar	not this year!

Practice 5 (continued)

8. Create a query to print the last names and number of years of service for each employee. If the employee has been employed for five or more years, print `5 years of service`. If the employee has been employed for 10 or more years, print `10 years of service`. If the employee has been employed for 15 or more years, print `15 years of service`. If none of these conditions match, print `maybe next year!` Sort the results by the `HIRE_DATE` column. Use the `EMPLOYEES` table.
Hint: Use `CASE` expressions and `TO_YMINTERVAL`.

	LAST_NAME	HIRE_DATE	SYSDATE	Awards
1	King	17-JUN-1987	18-NOV-2008	15 years of service
2	Kochhar	21-SEP-1989	18-NOV-2008	15 years of service
3	De Haan	13-JAN-1993	18-NOV-2008	15 years of service
4	Hunold	03-JAN-1990	18-NOV-2008	15 years of service
5	Ernst	21-MAY-1991	18-NOV-2008	15 years of service
6	Austin	25-JUN-1997	18-NOV-2008	10 years of service
7	Pataballa	05-FEB-1998	18-NOV-2008	10 years of service
8	Lorentz	07-FEB-1999	18-NOV-2008	5 years of service
9	Greenberg	17-AUG-1994	18-NOV-2008	10 years of service
10	Faviet	16-AUG-1994	18-NOV-2008	10 years of service
11	Chen	28-SEP-1997	18-NOV-2008	10 years of service
12	Sciarra	30-SEP-1997	18-NOV-2008	10 years of service
13	Urman	07-MAR-1998	18-NOV-2008	10 years of service
14	Popp	07-DEC-1999	18-NOV-2008	5 years of service
15	Raphaely	07-DEC-1994	18-NOV-2008	10 years of service

▪▪▪

94	Everett	03-MAR-1997	18-NOV-2008	10 years of service
95	McCain	01-JUL-1998	18-NOV-2008	10 years of service
96	Jones	17-MAR-1999	18-NOV-2008	5 years of service
97	Walsh	24-APR-1998	18-NOV-2008	10 years of service
98	Feeney	23-MAY-1998	18-NOV-2008	10 years of service
99	OConnell	21-JUN-1999	18-NOV-2008	5 years of service
100	Grant	13-JAN-2000	18-NOV-2008	5 years of service
101	Whalen	17-SEP-1987	18-NOV-2008	15 years of service
102	Hartstein	17-FEB-1996	18-NOV-2008	10 years of service
103	Fay	17-AUG-1997	18-NOV-2008	10 years of service
104	Mavris	07-JUN-1994	18-NOV-2008	10 years of service
105	Baer	07-JUN-1994	18-NOV-2008	10 years of service
106	Higgins	07-JUN-1994	18-NOV-2008	10 years of service
107	Gietz	07-JUN-1994	18-NOV-2008	10 years of service