PL/SQL - DATE & TIME

PL/SQL provides two classes of date and time related data types:

- Datetime data types
- Interval data types

The Datetime data types are:

- DATE
- TIMESTAMP
- TIMESTAMP WITH TIME ZONE
- TIMESTAMP WITH LOCAL TIME ZONE

The Interval data types are:

- INTERVAL YEAR TO MONTH
- INTERVAL DAY TO SECOND

Field Values for Datetime and Interval Data Types

Both **datetime** and **interval** data types consist of **fields**. The values of these fields determine the value of the datatype. The following table lists the fields and their possible values for datetimes and intervals.

Field Name	Valid Datetime Values	Valid Interval Values
YEAR	-4712 to 9999 (excluding year 0)	Any nonzero integer
MONTH	01 to 12	0 to 11
DAY	01 to 31 (limited by the values of MONTH and YEAR, according to the rules of the calendar for the locale)	Any nonzero integer
HOUR	00 to 23	0 to 23
MINUTE	00 to 59	0 to 59
SECOND	00 to 59.9(n), where 9(n) is the precision of time fractional seconds The 9(n) portion is not applicable for DATE.	0 to 59.9(n), where 9(n) is the precision of interval fractional seconds
TIMEZONE_HOUR	-12 to 14 (range accommodates daylight savings time changes) Not applicable for DATE or TIMESTAMP.	Not applicable
TIMEZONE_MINUTE	00 to 59 Not applicable for DATE or TIMESTAMP.	Not applicable
TIMEZONE_REGION	Not applicable for DATE or TIMESTAMP.	Not applicable
TIMEZONE_ABBR	Not applicable for DATE or TIMESTAMP.	Not applicable

The Datetime Data Types and Functions

Following are the Datetime data types:

- **DATE** - it stores date and time information in both character and number datatypes. It is made of information on century, year, month, date, hour, minute, and second. It is specified as:
- **TIMESTAMP** - it is an extension of the DATE datatype. It stores the year, month, and day of the DATE datatype, along with hour, minute, and second values. It is useful for storing precise time values.
- **TIMESTAMP WITH TIME ZONE** - it is a variant of TIMESTAMP that includes a time zone region name or a time zone offset in its value. The time zone offset is the difference (in hours and minutes) between local time and UTC. This datatype is useful for collecting and evaluating date information across geographic regions.
- **TIMESTAMP WITH LOCAL TIME ZONE** - it is another variant of TIMESTAMP that includes a time zone offset in its value.

Following table provides the Datetime functions (where, x has datetime value):

S.N	Function Name & Description
1	**ADD_MONTHS(x, y);** Adds y months to x.
2	**LAST_DAY(x);** Returns the last day of the month.
3	**MONTHS_BETWEEN(x, y);** Returns the number of months between x and y.
4	**NEXT_DAY(x, day);** Returns the datetime of the next *day* after x.
5	**NEW_TIME;** Returns the time/day value from a time zone specified by the user.
6	**ROUND(x [, unit]);** Rounds x;
7	**SYSDATE();** Returns the current datetime.
8	**TRUNC(x [, unit]);** Truncates x.

Timestamp functions (where, x has a timestamp value):

S.N	Function Name & Description
1	**CURRENT_TIMESTAMP();** Returns a TIMESTAMP WITH TIME ZONE containing the current session time along with the session time zone.
2	**EXTRACT({ YEAR | MONTH | DAY | HOUR | MINUTE | SECOND } | { TIMEZONE_HOUR | TIMEZONE_MINUTE } | { TIMEZONE_REGION | } TIMEZONE_ABBR) FROM x)** Extracts and returns a year, month, day, hour, minute, second, or time zone from x;
3	**FROM_TZ(x, time_zone);** Converts the TIMESTAMP x and time zone specified by time_zone to a TIMESTAMP WITH TIMEZONE.

4	**LOCALTIMESTAMP();** Returns a TIMESTAMP containing the local time in the session time zone.
5	**SYSTIMESTAMP();** Returns a TIMESTAMP WITH TIME ZONE containing the current database time along with the database time zone.
6	**SYS_EXTRACT_UTC(x);** Converts the TIMESTAMP WITH TIMEZONE x to a TIMESTAMP containing the date and time in UTC.
7	**TO_TIMESTAMP(x, [format]);** Converts the string x to a TIMESTAMP.
8	**TO_TIMESTAMP_TZ(x, [format]);** Converts the string x to a TIMESTAMP WITH TIMEZONE.

Examples:

The following code snippets illustrate the use of the above functions:

```
SELECT SYSDATE FROM DUAL;
```

Output:

```
08/31/2012 5:25:34 PM
```

```
SELECT TO_CHAR(CURRENT_DATE, 'DD-MM-YYYY HH:MI:SS') FROM DUAL;
```

Output:

```
31-08-2012 05:26:14
```

```
SELECT ADD_MONTHS(SYSDATE, 5) FROM DUAL;
```

Output:

```
01/31/2013 5:26:31 PM
```

```
SELECT LOCALTIMESTAMP FROM DUAL;
```

Output:

```
8/31/2012 5:26:55.347000 PM
```

The Interval Data Types and Functions

Following are the Interval data types:

- INTERVAL YEAR TO MONTH - it stores a period of time using the YEAR and MONTH datetime fields.
- INTERVAL DAY TO SECOND - it stores a period of time in terms of days, hours, minutes, and seconds.

Interval functions:

S.N	Function Name & Description
1	**NUMTODSINTERVAL(x, interval_unit);** Converts the number x to an INTERVAL DAY TO SECOND.
2	**NUMTOYMINTERVAL(x, interval_unit);**

	Converts the number x to an INTERVAL YEAR TO MONTH.
3	**TO_DSINTERVAL(x);** Converts the string x to an INTERVAL DAY TO SECOND.
4	**TO_YMINTERVAL(x);** Converts the string x to an INTERVAL YEAR TO MONTH.