**SHLOK ANAND**

**23070521142**

**Practical 3:**

**Part 5:**

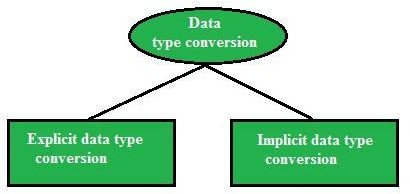
Conversion Function in SQL

In SQL **data type conversion** is important for effective **database management** and accurate query results. Data type conversion ensures that data from different sources or columns can be correctly interpreted and manipulated, especially when dealing with different formats like **numbers**, text, **dates**, and other data types.

**Types of Data Type Conversion in SQL**

There are two main types of data type conversion in SQL.

* **Implicit Data Type Conversion:** This is done automatically by the database management system (**DBMS**) when SQL operations involve columns of different data types. For instance, a **string** value might automatically be converted into a **numeric type** if required by a mathematical operation.
* **Explicit Data Type Conversion:** This is done by the user, who specifies the conversion. This is necessary when SQL cannot automatically convert between data types, or when more control over the conversion is needed.



# Overview of Conversion Functions

|  |  |  |  |
| --- | --- | --- | --- |
| **Function** | **Oracle (SQL\*Plus)** | **MySQL** | **Description** |
| TO\_CHAR() | Yes | ❌ No | Converts a date/number to a string |
| TO\_DATE() | Yes | ❌ No | Converts a string to a date |
| TO\_NUMBER() | Yes | ❌ No | Converts a string to a number |
| CAST() | Yes | Yes | Converts from one data type to another |
| CONVERT() | ❌ No | Yes | Converts string from one character set to another |
| FORMAT() | ❌ No | Yes | Formats numbers with decimal places |
| STR\_TO\_DATE () | ❌ No | Yes | Converts a string to a date |
| DATE\_FORMAT () | ❌ No | Yes | Formats a date as a string |
| TIME\_FORMAT () | ❌ No | Yes | Formats time values |

|  |  |  |  |
| --- | --- | --- | --- |
| UNIX\_TIMEST AMP() | ❌ No | Yes | Converts a date to Unix timestamp |
| FROM\_UNIXTI ME() | ❌ No | Yes | Converts Unix timestamp to a date |

1. **Conversion Functions in SQL\*Plus (Oracle) /skip if you want to use mysql platform**

Oracle provides TO\_CHAR(), TO\_DATE(), TO\_NUMBER(), and CAST() for conversion.

* 1. **TO\_CHAR() – Convert Date/Number to String**

**Use Case:** Format **date & time** into a human-readable string.

### SELECT TO\_CHAR(SYSDATE, 'YYYY-MM-DD HH24:MI:SS') AS

formatted\_date FROM dual;

## Output Example:

formatted\_date

2025-01-29 14:35:50

## Format Number as Currency:

SELECT TO\_CHAR(12345.67, 'L99,999.99') AS formatted\_currency FROM dual;

## Output Example:

formatted\_currency

$12,345.67

* 1. **TO\_DATE() – Convert String to Date**

**Use Case:** Convert a **string** into a **date format**.

SELECT TO\_DATE('2025-01-29', 'YYYY-MM-DD') AS converted\_date

FROM dual;

## Output Example:

converted\_date

29-JAN-25

## Using Different Date Formats:

SELECT TO\_DATE('29-01-2025', 'DD-MM-YYYY') FROM dual;

Sample output

* 1. **TO\_NUMBER() – Convert String to Number**

**Use Case:** Convert a **string** containing numbers into a **numeric type**.

SELECT TO\_NUMBER('12345.67') AS number\_value FROM dual;

## Output Example:

number\_value

12345.67

* 1. **CAST() – Convert Data Types**

**Use Case:** Convert a number to a string or vice versa.

SELECT CAST(123.45 AS VARCHAR2(10)) AS string\_value FROM

dual;

## Output Example:

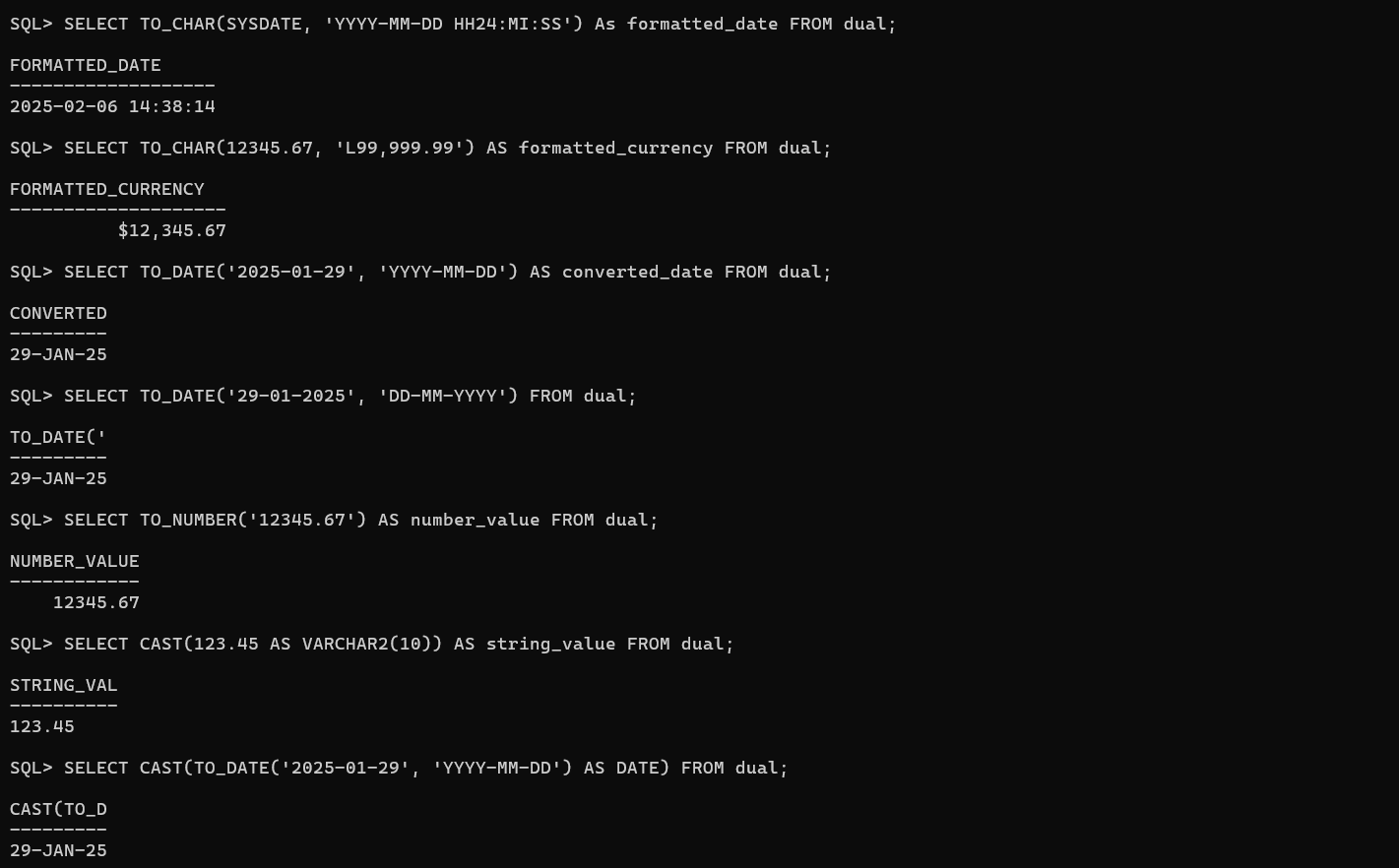
string\_value

123.45

## Convert String to Date:

## SELECT CAST(TO\_DATE('2025-01-29', 'YYYY-MM-DD') AS DATE)

FROM dual;



# 

# Real-World Use Cases of Conversion Functions

**Financial Data Reporting**

Convert salary figures into **formatted currency**.

SELECT emp\_id, TO\_CHAR(salary, 'L99,999.99') AS formatted\_salary FROM employees;

**Log Analysis (MySQL)**

Convert timestamps into **human-readable format**.

SELECT FROM\_UNIXTIME(UNIX\_TIMESTAMP()) AS current\_time;

**Data Migration**

When migrating from **CSV files**, convert **strings to dates**.

SELECT STR\_TO\_DATE('29-01-2025', '%d-%m-%Y') AS

converted\_date;

# Summary Table

## Function Oracle (SQL\*Plus)

## MySQ L

## Purpose

TO\_CHAR() Yes ❌ No Convert date/number to string

TO\_DATE() Yes ❌ No Convert string to date

|  |  |  |  |
| --- | --- | --- | --- |
| TO\_NUMBER(  ) | Yes | ❌ No | Convert string to number |
| CAST() | Yes | Yes | Convert between data types |
| CONVERT() | ❌ No | Yes | Convert between character sets |
| FORMAT() | ❌ No | Yes | Format number with commas |
| STR\_TO\_DAT | ❌ No | Yes | Convert string to date |
| E() |  |  |  |
| DATE\_FORMA | ❌ No | Yes | Format a date as a string |
| T() |  |  |  |
| TIME\_FORMA | ❌ No | Yes | Format time values |
| T() |  |  |  |
| UNIX\_TIMES | ❌ No | Yes | Convert date to Unix |

### TAMP()

FROM\_UNIXT IME()

timestamp

❌ No Yes Convert Unix timestamp to date

# Advanced Real-World Use Cases of Conversion Functions in MySQL & SQL\*Plus (Oracle)

**E-Commerce: Converting Prices for Different Currencies**



**1**

**Scenario:** An e-commerce site needs to convert prices from USD to INR and format them properly.

## Oracle (SQL\*Plus):

### SELECT

product\_id, product\_name,

TO\_CHAR(price\_usd \* 83.50, 'L99,999.99') AS price\_inr FROM products;

## MySQL:

### SELECT

product\_id, product\_name,

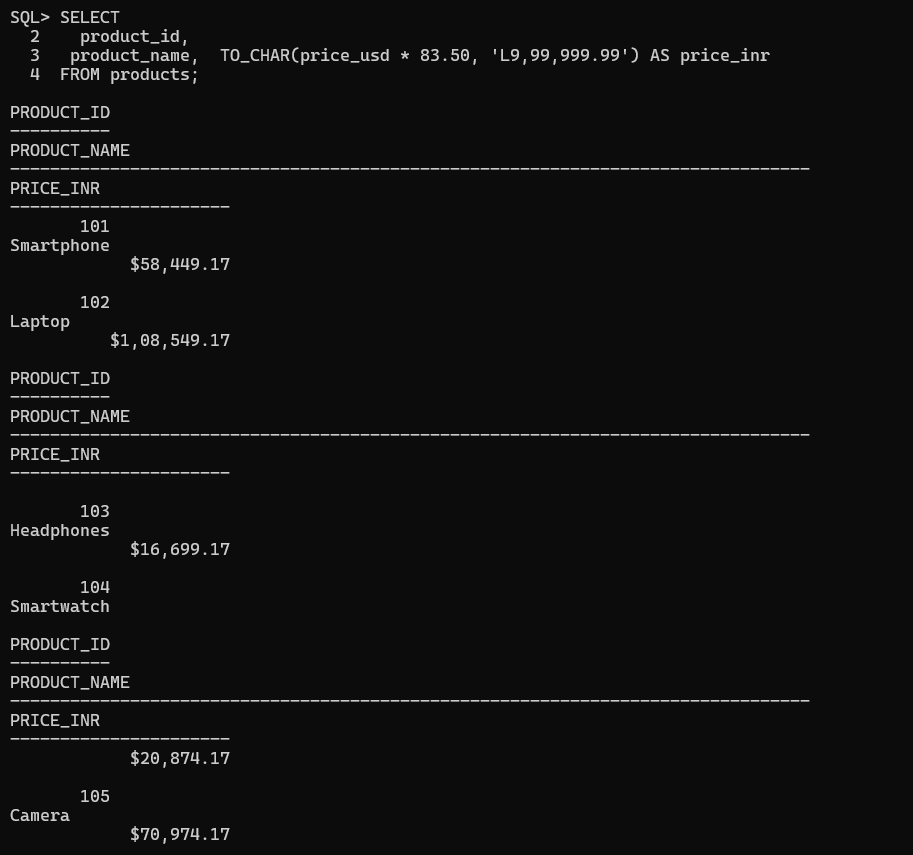
FORMAT(price\_usd \* 83.50, 2) AS price\_inr FROM products;

## Why?

* Uses TO\_CHAR() in Oracle and FORMAT() in MySQL to **add currency formatting**.
* 1 USD = **83.50 INR** (exchange rate example).

**Example Output:**

|  |  |  |
| --- | --- | --- |
| **product\_id** | **product\_name** | **price\_inr** |
| 101 | iPhone 15 | ₹99,999.99 |
| 202 | MacBook Pro | ₹2,19,999.99 |

****

# Banking: Detecting Fraudulent Transactions Using Date Conversions



**2**

**Scenario:** A bank flags **suspicious transactions** that happened at **odd hours (midnight to 4 AM)**.

## Oracle (SQL\*Plus):

SELECT transaction\_id, account\_id, amount, TO\_CHAR(transaction\_time, 'HH24:MI') AS transaction\_hour FROM transactions

WHERE EXTRACT(HOUR FROM transaction\_time) BETWEEN 0 AND 4;

## MySQL:

SELECT transaction\_id, account\_id, amount, TIME\_FORMAT(transaction\_time, '%H:%i') AS transaction\_hour FROM transactions

WHERE HOUR(transaction\_time) BETWEEN 0 AND 4;

## Why?

* Uses TO\_CHAR() (Oracle) and TIME\_FORMAT() (MySQL) to **extract and format time**.
* Filters transactions **between 00:00 and 04:00**.

**Example Output:**

|  |  |  |  |
| --- | --- | --- | --- |
| **transaction\_id** | **account\_id** | **amount** | **transaction\_hour** |
| 89234 | 123456 | 5000 | 02:30 |
| 97345 | 789012 | 25000 | 03:15 |

# IoT & Smart Devices: Storing and Retrieving Unix Timestamps



**3**

**Scenario:** A smart home system stores **sensor readings** as Unix timestamps and needs human-readable timestamps.

## Oracle (SQL\*Plus) - Convert Unix Timestamp to Readable Date:

SELECT sensor\_id, FROM\_TZ(TO\_TIMESTAMP(1706505600), 'UTC')

AS reading\_time FROM sensor\_logs;

## MySQL:

SELECT sensor\_id, FROM\_UNIXTIME(1706505600) AS reading\_time FROM sensor\_logs;

## Why?

* Converts 1706505600 (Unix timestamp) into a **readable date-time format**.

**Example Output:**

**reading\_time**

**sensor\_id**

2025-01-29 12:00:00

101

# Marketing Analytics: Extracting Month and Year from Dates



**4**

**Scenario:** A company wants to analyze customer purchases by **month and year**.

## Oracle (SQL\*Plus):

### SELECT

customer\_id, purchase\_date,

TO\_CHAR(purchase\_date, 'Month') AS purchase\_month, TO\_CHAR(purchase\_date, 'YYYY') AS purchase\_year

FROM purchases;

## MySQL:

### SELECT

customer\_id, purchase\_date,

DATE\_FORMAT(purchase\_date, '%M') AS purchase\_month, DATE\_FORMAT(purchase\_date, '%Y') AS purchase\_year

FROM purchases;

## Why?

* Uses TO\_CHAR() (Oracle) and DATE\_FORMAT() (MySQL) to extract

**month and year** from a **purchase date**.

## Example Output:

**customer\_id purchase\_date purchase\_month purchase\_year**

501 2025-01-29 January 2025

# Data Migration: Converting String Dates into Proper Date Format



**5**

**Scenario:** A company migrating old **CSV data** where dates are stored as strings (DD/MM/YYYY).

## Oracle (SQL\*Plus):

SELECT TO\_DATE('29/01/2025', 'DD/MM/YYYY') AS formatted\_date FROM dual;

## MySQL:

SELECT STR\_TO\_DATE('29/01/2025', '%d/%m/%Y') AS

formatted\_date;

## Why?

* Converts 29/01/2025 (string) into a **date type** in Oracle (TO\_DATE()) and MySQL (STR\_TO\_DATE()).

**Example Output:**

2025-01-29

**formatted\_date**

# Logistics & Delivery: Calculating Expected Delivery Time Based on Distance



**6**

**Scenario:** Estimate delivery **ETA** based on **distance traveled** and **average speed**.

## Oracle (SQL\*Plus):

### SELECT

order\_id, distance\_km,

ROUND(distance\_km / 60, 2) AS estimated\_hours FROM deliveries;

## MySQL:

### SELECT

order\_id, distance\_km,

FORMAT(distance\_km / 60, 2) AS estimated\_hours FROM deliveries;

## Why?

* Divides distance\_km by 60 km/h (average speed).

**Example Output:**

|  |  |  |
| --- | --- | --- |
| **order\_id** | **distance\_km** | **estimated\_hours** |
| 1001 | 120 | 2.00 |

# Social Media Analytics: Converting Post Dates into Readable Formats



**7**

**Scenario:** A social media platform needs to display post timestamps **beautifully**. **Oracle (SQL\*Plus):**

SELECT post\_id, TO\_CHAR(post\_date, 'Month DD, YYYY HH24:MI') AS formatted\_date FROM posts;

## MySQL:

SELECT post\_id, DATE\_FORMAT(post\_date, '%M %d, %Y %H:%i') AS formatted\_date FROM posts;

**Why?**

* Converts **date into a social-media friendly format**.

**Example Output:**

|  |  |
| --- | --- |
| **post\_id** | **formatted\_date** |
| 555 | January 29, 2025 14:35 |

**Summary Table**

|  |  |  |
| --- | --- | --- |
| **Scenario** | **Oracle (SQL\*Plus)** | **MySQL** |
| Convert prices to INR | TO\_CHAR(price, 'L99,999.99') | FORMAT(price, 2) |
| Detect fraud based on time | EXTRACT(HOUR FROM  transaction\_time) | HOUR(transaction\_tim e) |
| Convert Unix timestamp | FROM\_TZ(TO\_TIMESTAMP(  ...), 'UTC') | FROM\_UNIXTIME(...) |
| Extract month & year | TO\_CHAR(date, 'Month YYYY') | DATE\_FORMAT(date, '%M %Y') |
| Convert string to date | TO\_DATE('29/01/2025', 'DD/MM/YYYY') | STR\_TO\_DATE('29/01/2 025', '%d/%m/%Y') |
| Estimate delivery ETA | ROUND(distance\_km / 60, 2) | FORMAT(distance\_km / 60, 2) |
| Format social media timestamps | TO\_CHAR(post\_date, 'Month DD, YYYY HH24:MI') | DATE\_FORMAT(post\_dat e, '%M %d, %Y  %H:%i') |