

Practical 3: Part

5: sushmit

partakke

23070521156

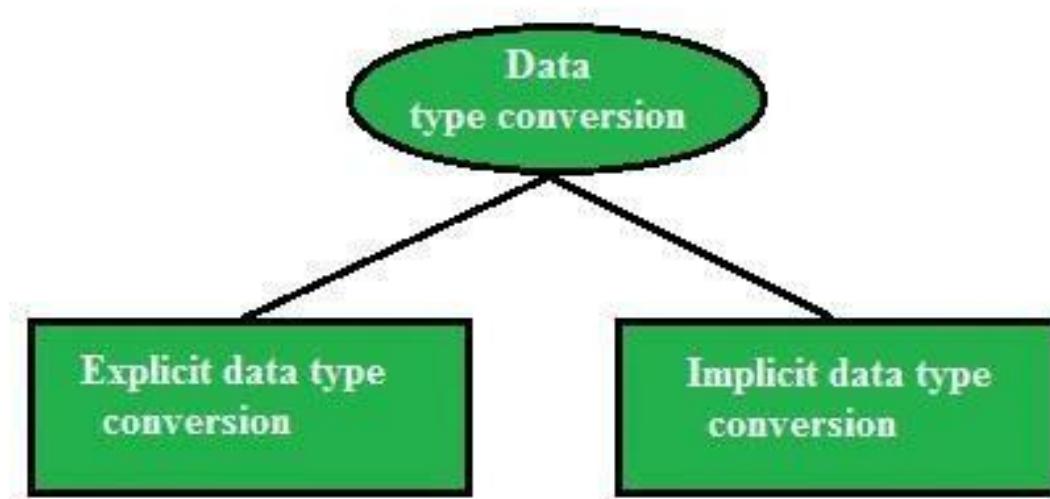
## Conversion Function in SQL

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.



## 1. 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

<code>TIME_FORMAT()</code>	✗ No	Yes	Formats time values
<code>UNIX_TIMESTAMP()</code>	✗ No	Yes	Converts a date to Unix timestamp
<code>FROM_UNIXTIME()</code>	✗ No	Yes	Converts Unix timestamp to a date

## 2. 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.

### 2.1 `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
```

---

## **2.2 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;
```

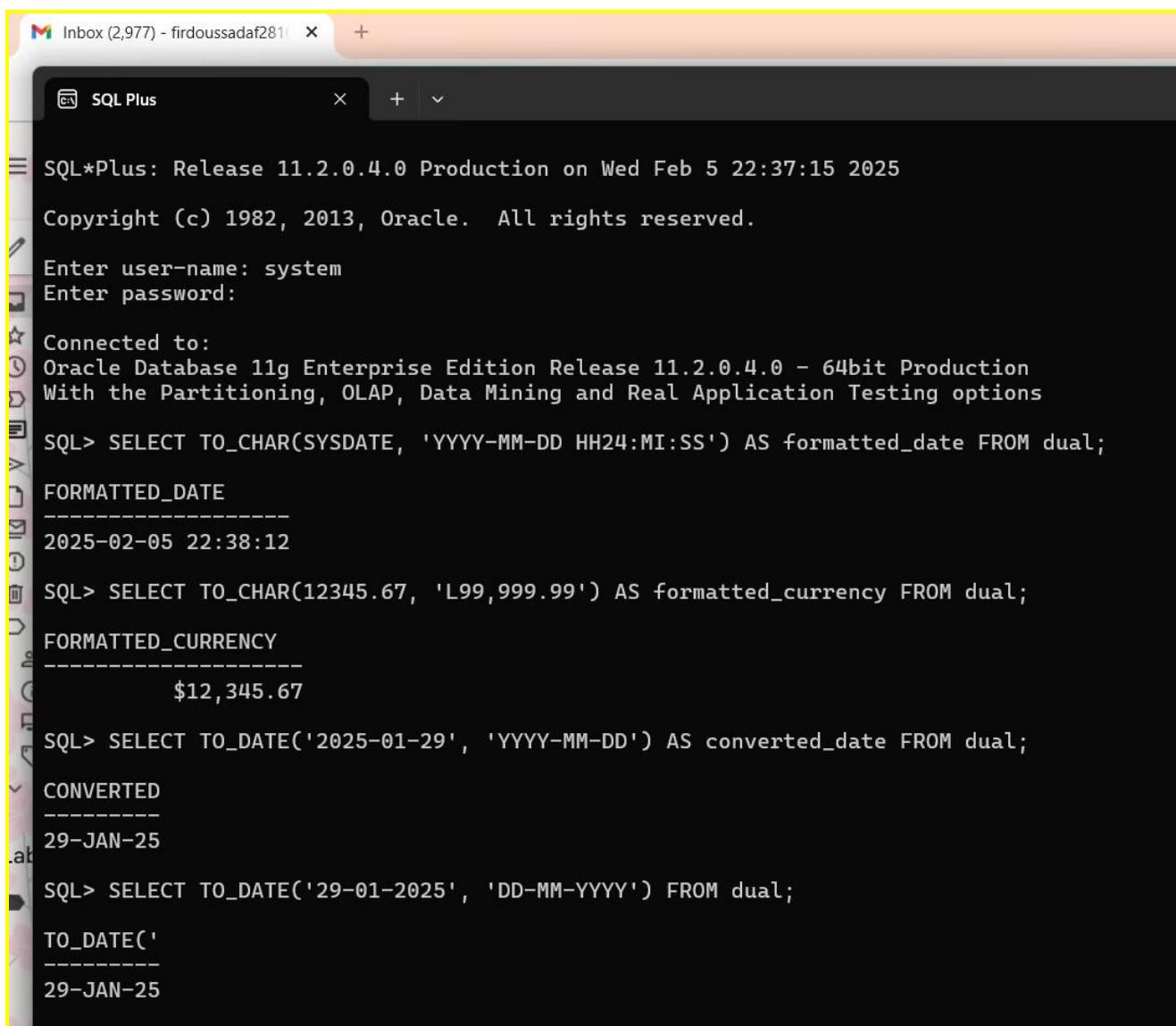
## **Output Example:**

```
converted_date -----  
29-JAN-25
```

## **Using Different Date Formats:**

```
SELECT TO_DATE('29-01-2025', 'DD-MM-YYYY') FROM dual;
```

### Sample output



The screenshot shows a terminal window titled "SQL Plus" running on an Oracle database. The session starts with the standard Oracle welcome message and prompts for a user name and password. It then connects to the database and executes several SQL commands:

- A query to select the current date formatted as YYYY-MM-DD HH24:MI:SS.
- A query to select a number (12345.67) formatted as currency (\$12,345.67).
- A query to select a date ('2025-01-29') converted to a standard date format.
- A query to select a date ('29-01-2025') converted to a standard date format.

The output shows the results of these queries, demonstrating the use of the TO\_DATE and TO\_CHAR functions.

## 2.3 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
```

---

## **2.4 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;
```



```
Oracle Database 11g Enterprise Edition Release 11.2.0.4.0 - 64bit Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options

SQL> SELECT TO_CHAR(SYSDATE, 'YYYY-MM-DD HH24:MI:SS') AS
  2 formatted_date FROM dual;

FORMATTED_DATE
-----
2025-02-06 14:17:18

SQL> SELECT TO_CHAR(12345.67, 'L99,999.99') AS formatted_currency
  2 FROM dual;

FORMATTED_CURRENCY
-----
$12,345.67

SQL> SELECT TO_DATE('2025-01-29', 'YYYY-MM-DD') AS converted_date
  2 FROM dual;

CONVERTED
-----
29-JAN-25

SQL> SELECT TO_NUMBER('12345.6211111111111117') AS number_value FROM dual;

NUMBER_VALUE
-----
12345.6211

SQL> SELECT CAST(123.45!!!! AS VARCHAR2(10)) AS string_value FROM dual;
SELECT CAST(123.45!!!! AS VARCHAR2(10)) AS string_value FROM dual
*
ERROR at line 1:
ORA-00905: missing keyword

SQL> SELECT CAST(123 AS VARCHAR2(10)) AS string_value FROM dual;

STRING_VAL
-----
123

SQL> SELECT CAST(TO_DATE('2025-02-06', 'YYYY-MM-DD') AS DATE)
  2 FROM dual;

CAST(TO_D
-----
06-FEB-25
```

### 3. Conversion Functions in MySQL //SKIP IF DONE WITH ORACLE SQLPLUS

MySQL provides `CAST()`, `CONVERT()`, `STR_TO_DATE()`, `DATE_FORMAT()`, etc.

#### 3.1 `CAST()` – Convert Data Types

**Use Case:** Convert an integer to a **string**.

```
SELECT CAST(12345 AS CHAR) AS string_value;
```

**Output Example:**

diff

```
string_value -----
12345
```

**Convert a String to an Integer:**

```
SELECT CAST('12345' AS SIGNED) AS number_value;
```

---

#### 3.2 `CONVERT()` – Convert Between Character Sets

**Use Case:** Change **character encoding**.

```
SELECT CONVERT('Héllö' USING utf8mb4) AS utf8_text;
```

### Convert a Number to String:

```
SELECT CONVERT(12345, CHAR) AS string_value;
```

---

### 3.3 FORMAT() – Format Number with Commas

**Use Case:** Display large numbers with commas.

```
SELECT FORMAT(1234567.89, 2) AS formatted_number;
```

### Output Example:

diff

```
formatted_number
```

```
-----
```

```
1,234,567.89
```

---

### 3.4 STR\_TO\_DATE() – Convert String to Date

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

```
SELECT STR_TO_DATE('29-01-2025', '%d-%m-%Y') AS converted_date;
```

### **Output Example:**

diff

```
converted_date -----
2025-01-29
```

---

### **3.5 DATE\_FORMAT() – Format a Date as a String**

**Use Case:** Display **formatted dates**.

```
SELECT DATE_FORMAT(NOW(), '%W, %M %d, %Y') AS formatted_date;
```

### **Output Example:**

diff

```
formatted_date
-----
Tuesday, January 29, 2025
```

---

### **3.6 TIME\_FORMAT() – Format Time**

**Use Case:** Convert **24-hour time** into **12-hour format**.

```
SELECT TIME_FORMAT('14:35:50', '%h:%i %p') AS formatted_time;
```

### **Output Example:**

diff

```
formatted_time -----
02:35 PM
```

---

### **3.7 UNIX\_TIMESTAMP() – Convert Date to Unix Timestamp**

**Use Case:** Store dates as **timestamps**.

```
SELECT UNIX_TIMESTAMP('2025-01-29 14:35:50') AS unix_time;
```

### **Output Example:**

```
unix_time -----
1740792950
```

### **3.8 FROM\_UNIXTIME() – Convert Unix Timestamp to Date**

**Use Case:** Convert **timestamps** back to a **date**.

```
SELECT FROM_UNIXTIME(1740792950) AS converted_date;
```

## Output Example:

```
converted_date -----
2025-01-29 14:35:50
```

## 4. 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;
```

```
SQL> SELECT emp_id, TO_CHAR(salary, 'L99,999.99') AS
2 formatted_salary FROM employees;
```

EMP_ID	FORMATTED_SALARY
1	\$50,000.00
2	\$75,000.75
3	\$95,000.00
4	#####
5	#####

### Log Analysis (MySQL)

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

```
SQL> SELECT TO_CHAR(SYSDATE, 'DD-MON-YYYY HH24:MI:SS') AS current_time  
  2  FROM dual;  
  
CURRENT_TIME  
-----  
06-FEB-2025 14:52:22
```

```
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;
```

```
SQL> SELECT TO_DATE('29-01-2025', 'DD-MM-YYYY') AS converted_date  
  2  FROM dual;  
  
CONVERTED  
-----  
29-JAN-25
```

## 5. Summary Table

Function	Oracle (SQL*Plus)	MySQL	Purpose
TO_CHAR()	Yes	X	No Convert date/number to string

<code>TO_DATE()</code>	Yes	<b>✗</b> No	Convert string to date
<code>TO_NUMBER()</code>	Yes	<b>✗</b> No	Convert string to number
<code>CAST()</code>	Yes	Yes	Convert between data types
<code>CONVERT()</code>	<b>✗</b> No	Yes	Convert between character sets
<code>FORMAT()</code>	<b>✗</b> No	Yes	Format number with commas
<code>STR_TO_DATE()</code>	<b>✗</b> No	Yes	Convert string to date
<code>DATE_FORMAT()</code>	<b>✗</b> No	Yes	Format a date as a string
<code>TIME_FORMAT()</code>	<b>✗</b> No	Yes	Format time values
<code>UNIX_TIMESTAMP()</code>	<b>✗</b> No	Yes	Convert date to Unix timestamp
<code>FROM_UNIXTIME()</code>	<b>✗</b> No	Yes	Convert Unix timestamp to date

---

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

## 1 E-Commerce: Converting Prices for Different Currencies

```
SQL> SELECT
 2   product_id,
 3   product_name,
 4   TO_CHAR(price_usd * 83.50, 'L99,999.99') AS price_inr
 5 FROM products;

PRODUCT_ID
-----
PRODUCT_NAME
-----
PRICE_INR
-----
      1
Laptop          $66,800.00

      2
Smartphone      $41,750.00

PRODUCT_ID
-----
PRODUCT_NAME
-----
PRICE_INR
-----
      3
Headphones       $4,175.00

      4
Monitor

PRODUCT_ID
-----
PRODUCT_NAME
-----
PRICE_INR
-----
      $16,700.00

      5
Keyboard         $2,505.00
```

**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

## **2 Banking: Detecting Fraudulent Transactions Using Date Conversions**

```

SQL> CREATE TABLE transactions (
  2      transaction_id NUMBER PRIMARY KEY,
  3      account_id NUMBER,
  4      amount NUMBER(10,2),
  5      transaction_time TIMESTAMP
 6  );
Table created.

SQL> INSERT INTO transactions (transaction_id, account_id, amount, transaction_time)
  2  VALUES (1, 101, 500.00, TO_TIMESTAMP('2024-02-06 01:30:00', 'YYYY-MM-DD HH24:MI:SS'));

1 row created.

SQL>
SQL> INSERT INTO transactions (transaction_id, account_id, amount, transaction_time)
  2  VALUES (2, 102, 1200.00, TO_TIMESTAMP('2024-02-06 03:45:00', 'YYYY-MM-DD HH24:MI:SS'));

1 row created.

SQL>
SQL> INSERT INTO transactions (transaction_id, account_id, amount, transaction_time)
  2  VALUES (3, 103, 750.00, TO_TIMESTAMP('2024-02-06 06:15:00', 'YYYY-MM-DD HH24:MI:SS'));

1 row created.

SQL>
SQL> INSERT INTO transactions (transaction_id, account_id, amount, transaction_time)
  2  VALUES (4, 104, 200.00, TO_TIMESTAMP('2024-02-06 02:10:00', 'YYYY-MM-DD HH24:MI:SS'));

1 row created.

SQL>
SQL> INSERT INTO transactions (transaction_id, account_id, amount, transaction_time)
  2  VALUES (5, 105, 900.00, TO_TIMESTAMP('2024-02-06 04:50:00', 'YYYY-MM-DD HH24:MI:SS'));

1 row created.

SQL> SELECT transaction_id, account_id, amount,
  2  TO_CHAR(transaction_time, 'HH24:MI') AS transaction_hour
  3  FROM transactions
  4  WHERE EXTRACT(HOUR FROM transaction_time) BETWEEN 0 AND 4;

TRANSACTION_ID ACCOUNT_ID      AMOUNT TRANS
----- ----- -----
          1        101      500 01:30
          2        102     1200 03:45
          4        104      200 02:10
          5        105      900 04:50

```

SQL> |

**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

```
SQL> SELECT
  2      sensor_id,
  3      FROM_TZ(
  4          TO_TIMESTAMP('1970-01-01 00:00:00', 'YYYY-MM-DD HH24:MI:SS')
  5          + NUMTODSINTERVAL(reading_unix, 'SECOND'),
  6          'UTC'
  7      ) AS reading_time
  8  FROM sensor_logs;

SENSOR_ID
-----
READING_TIME
-----
1
29-JAN-24 05.20.00.000000000 AM UTC

2
29-JAN-24 09.06.40.000000000 AM UTC

3
29-JAN-24 12.53.20.000000000 PM UTC

SENSOR_ID
-----
READING_TIME
-----
4
29-JAN-24 04.40.00.000000000 PM UTC

5
29-JAN-24 08.26.40.000000000 PM UTC
```

**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:

sensor_id	reading_time
101	2025-01-29 12:00:00

## 4 Marketing Analytics: Extracting Month and Year from Dates

```
1 row created.  
  
SQL> SELECT  
  2      customer_id,  
  3      purchase_date,  
  4      TO_CHAR(purchase_date, 'Month') AS purchase_month,  
  5      TO_CHAR(purchase_date, 'YYYY') AS purchase_year  
  6  FROM purchases;  
  
CUSTOMER_ID PURCHASE_ PURCHASE_ PURC  
----- ----- -----  
    101 06-FEB-24 February  2024  
    102 15-JUL-23 July    2023  
    103 25-DEC-22 December 2022  
    104 10-MAY-21 May     2021  
    105 30-SEP-20 September 2020
```

**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

---

## 5 Data Migration: Converting String Dates into Proper Date Format

```
SQL> SELECT id, TO_CHAR(formatted_date, 'DD/MM/YYYY') AS formatted_date
  2  FROM date_table;

    ID FORMATTED_
-----
 1 29/01/2025
 2 15/03/2023
 3 08/12/2021
 4 22/06/2020
 5 10/10/2019
```

**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:**

formatted_date
2025-01-29

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

```
SQL> SELECT
  2      order_id,
  3      distance_km,
  4      ROUND(distance_km / 60, 2) AS estimated_hours
  5  FROM deliveries;

ORDER_ID DISTANCE_KM ESTIMATED_HOURS
-----  -----  -----
      1        120          2
      2        250         4.17
      3         80         1.33
      4        150          2.5
      5        200         3.33
```

**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:

<code>order_id</code>	<code>distance_km</code>	<code>estimated_hours</code>
1001	120	2.00

---

## Social Media Analytics: Converting Post Dates into Readable Formats

```

SQL> SELECT
2      post_id,
3      TO_CHAR(post_date, 'Month DD, YYYY HH24:MI') AS formatted_date
4  FROM posts;

POST_ID FORMATTED_DATE
-----
1 February 06, 2024 14:30
2 July      15, 2023 08:45
3 December 25, 2022 19:00
4 May       10, 2021 10:00
5 September 30, 2020 22:15

```

**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_time)
Convert Unix timestamp	FROM_TZ(TO_TIMESTAMP(...), 'UTC')	FROM_UNIXTIME(...)

Extract month & year	<code>TO_CHAR(date, 'Month YYYY')</code>	<code>DATE_FORMAT(date, '%M %Y')</code>
Convert string to date	<code>TO_DATE('29/01/2025', 'DD/MM/YYYY')</code>	<code>STR_TO_DATE('29/01/2025', '%d/%m/%Y')</code>
Estimate delivery ETA	<code>ROUND(distance_km / 60, 2)</code>	<code>FORMAT(distance_km / 60, 2)</code>
Format social media timestamps	<code>TO_CHAR(post_date, 'Month DD, YYYY HH24:MI')</code>	<code>DATE_FORMAT(post_date, '%M %d, %Y %H:%i')</code>