SHIVANSH LOHANI
23070521141

Practical 3:

Part 5:

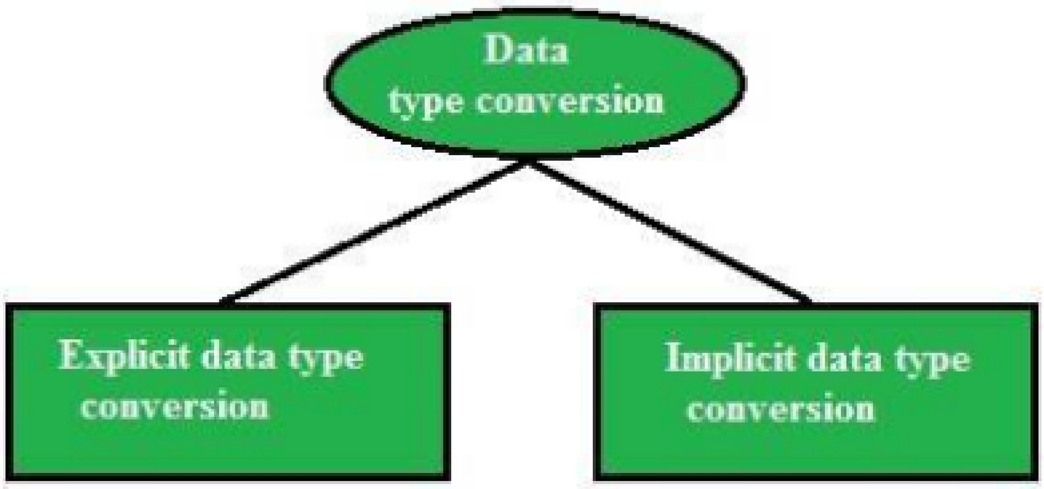
Conversion Function in SQL

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.



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
`TIME_FORMAT()`	❌ No	Yes	Formats time values

`UNIX_TIMESTAMP()`	❌ No	Yes	Converts a date to Unix timestamp
`FROM_UNIXTIME()`	❌ 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.

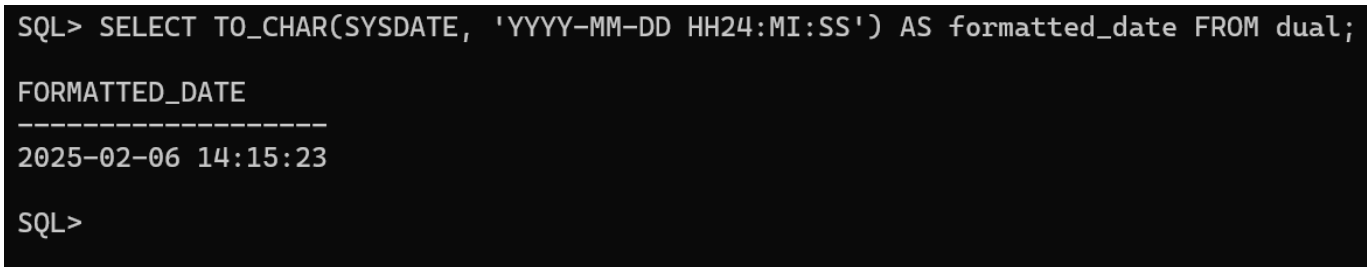
2.1 `TO_CHAR()` – Convert Date/Number to String

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:

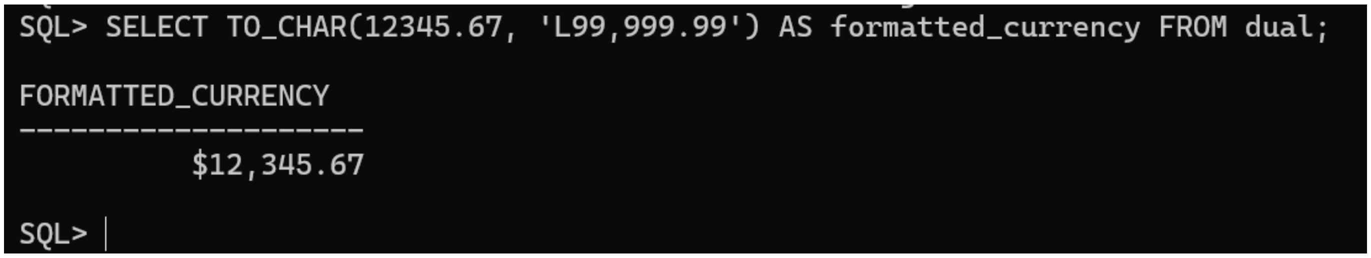
```
formatted_date
--------------------
2025-01-29 14:35:50
```



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

FORMATTED_DATE
-------------------
2025-02-06 14:15:23

SQL>
```

Format Number as Currency:

```
SELECT TO_CHAR(12345.67, 'L99,999.99') AS formatted_currency
FROM dual;
```



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

FORMATTED_CURRENCY
---------------------
          $12,345.67

SQL>
```

Output Example:

```
formatted_currency
------------------
$12,345.67
```

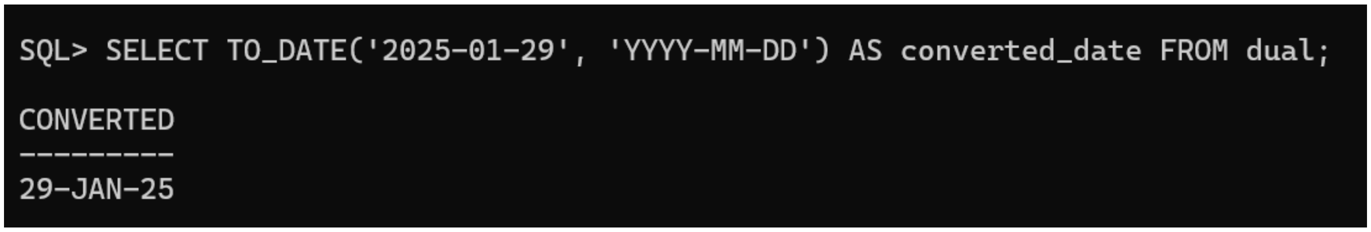
2.2 `TO_DATE()` – Convert String to Date

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:

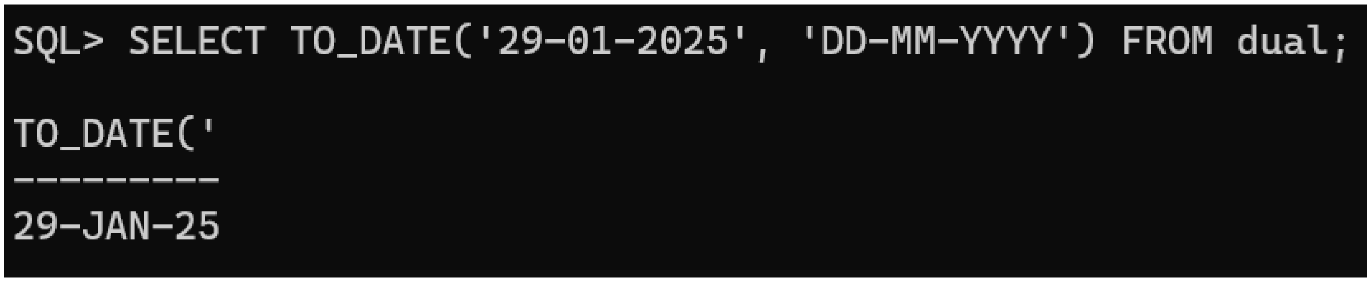
```
converted_date
---------------
29-JAN-25
```



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

CONVERTED
---------
29-JAN-25
```

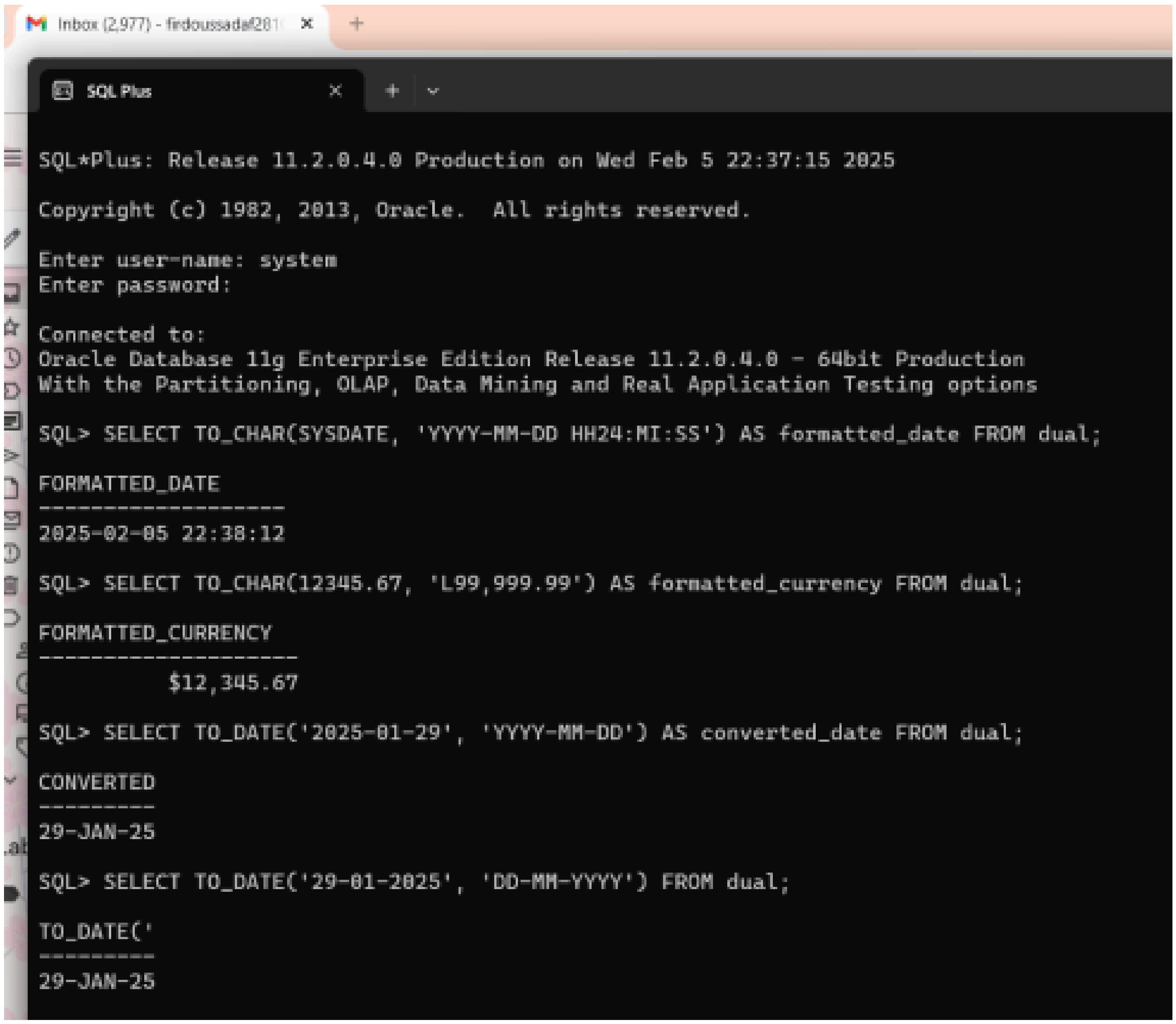
Using Different Date Formats:

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



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

TO_DATE('
---------
29-JAN-25
```

Sample output



```
SQL*Plus: Release 11.2.0.4.0 Production on Wed Feb 5 22:37:15 2025

Copyright (c) 1982, 2013, Oracle.  All rights reserved.

Enter user-name: system
Enter password:

Connected to:
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 formatted_date FROM dual;

FORMATTED_DATE
-------------------
2025-02-05 22:38:12

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

FORMATTED_CURRENCY
--------------------
          $12,345.67

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

CONVERTED
---------
29-JAN-25

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

TO_DATE('
---------
29-JAN-25
```

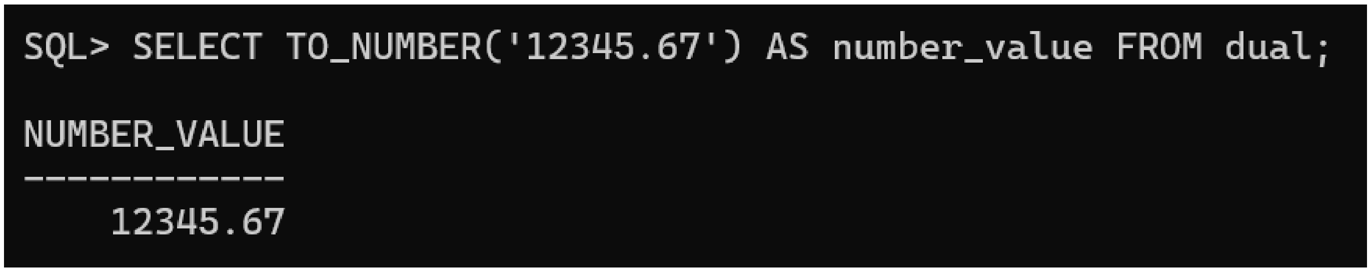
2.3 `TO_NUMBER()` – Convert String to Number

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

```
SELECT TO_NUMBER('12345.67') AS number_value FROM dual;
```

Output Example:

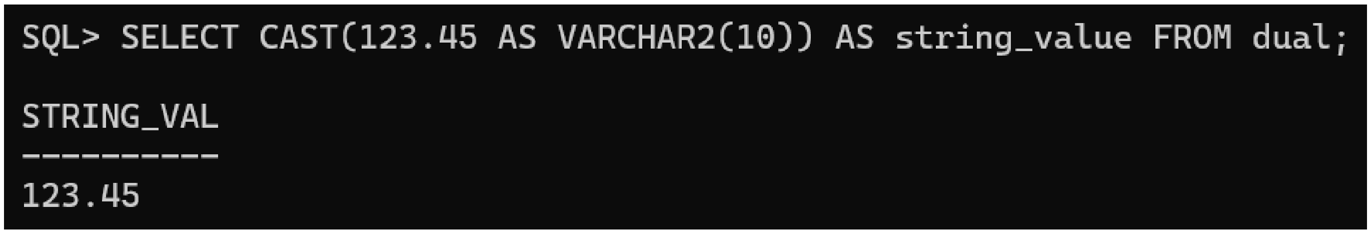
```
number_value

------------

12345.67
```


```
SQL> SELECT TO_NUMBER('12345.67') AS number_value FROM dual;

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;
```

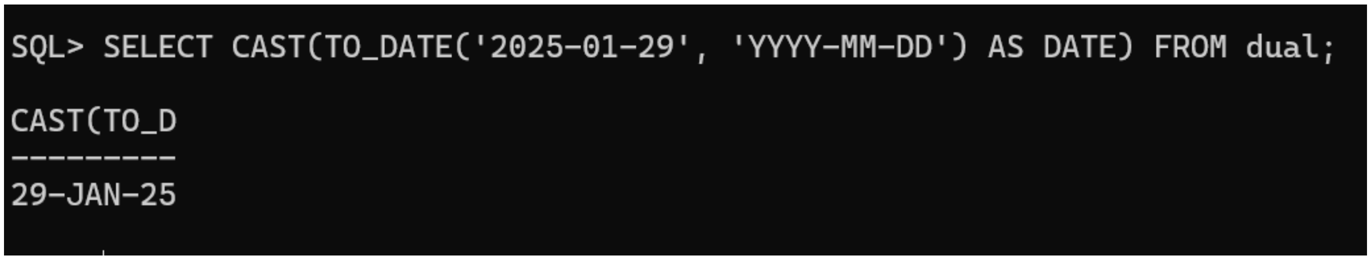

```
SQL> SELECT CAST(123.45 AS VARCHAR2(10)) AS string_value FROM dual;

STRING_VAL
----------
123.45
```

Output Example:

```
string_value
------------
123.45
```

Convert String to Date:

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



```
SQL> SELECT CAST(TO_DATE('2025-01-29', 'YYYY-MM-DD') AS DATE) FROM dual;

CAST(TO_D
---------
29-JAN-25
```

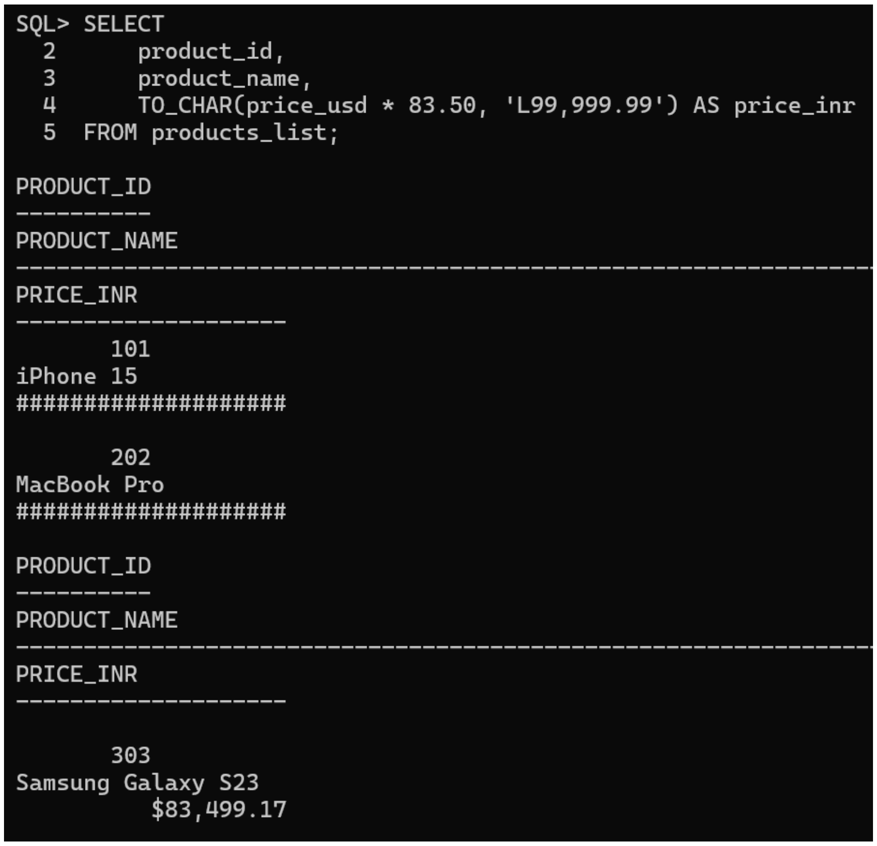
Advanced Real-World Use Cases of Conversion Functions in MySQL & SQL*Plus (Oracle)

1E-Commerce: Converting Prices for Different Currenci

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;
```



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

PRODUCT_ID
----------
PRODUCT_NAME
------------------------------------------------------------------
PRICE_INR
--------------------
       101
iPhone 15
####################

       202
MacBook Pro
####################

PRODUCT_ID
----------
PRODUCT_NAME
------------------------------------------------------------------
PRICE_INR
--------------------

       303
Samsung Galaxy S23
          $83,499.17
```

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:

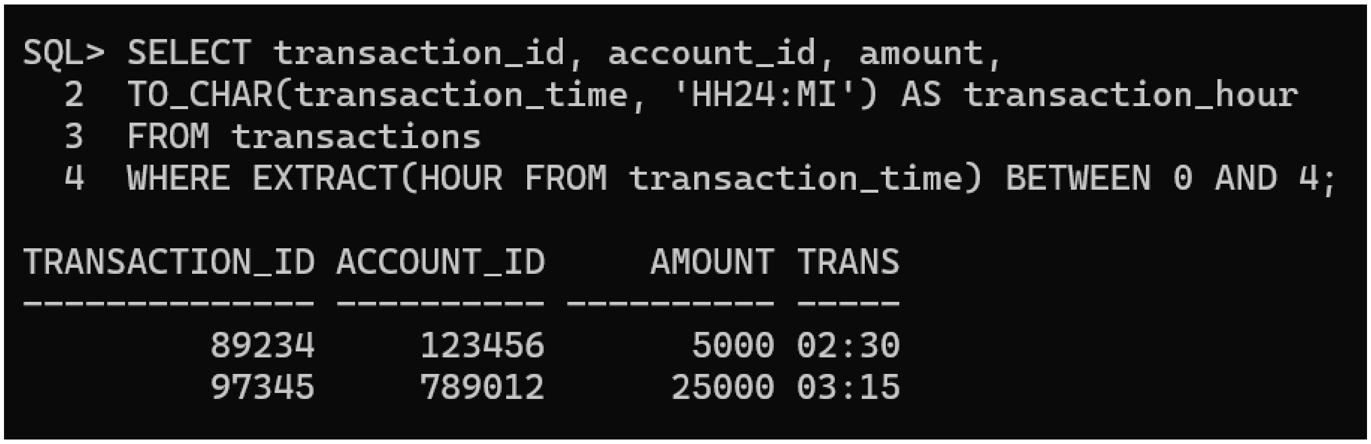
product_id	product_name	price_inr
101	iPhone 15	₹99,999.99
202	MacBook Pro	₹2,19,999.99

2Banking: Detecting Fraudulent Transactions Using Da Conversions

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;
```



```
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
-------------- ---------- ---------- -----
         89234     123456       5000 02:30
         97345     789012      25000 03:15
```

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

3IoT & Smart Devices: Storing and Retrieving Un Timestamps

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:

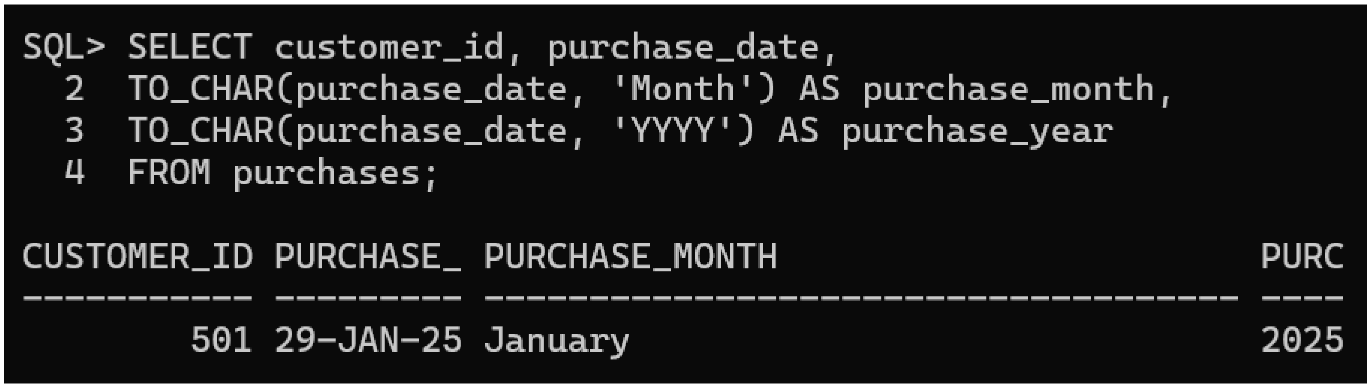
sensor_id	reading_time
101	2025-01-29 12:00:00

4️⃣Marketing Analytics: Extracting Month and Year fr Dates

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;
```



```
SQL> SELECT customer_id, purchase_date,
  2  TO_CHAR(purchase_date, 'Month') AS purchase_month,
  3  TO_CHAR(purchase_date, 'YYYY') AS purchase_year
  4  FROM purchases;

CUSTOMER_ID PURCHASE_ PURCHASE_MONTH                       PURC
----------- --------- ------------------------------------ ----
        501 29-JAN-25 January                              2025
```

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

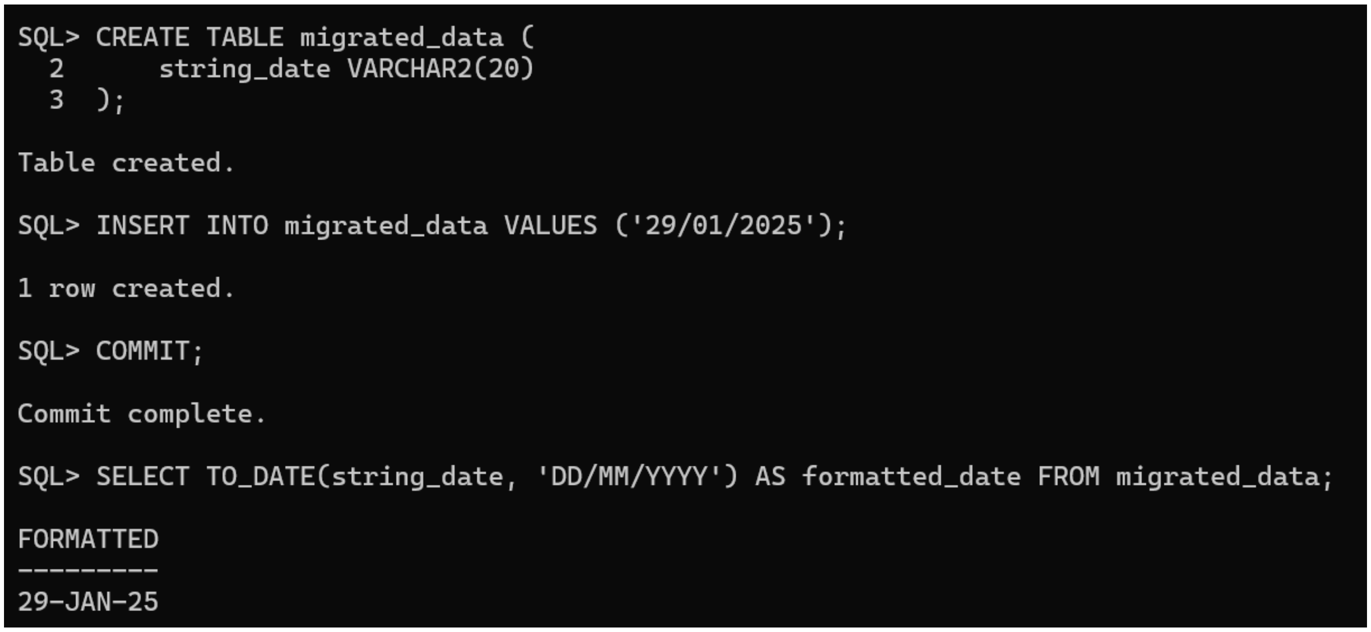
2025-01-29 January 2025

5️⃣Data Migration: Converting String Dates into Proper Da Format

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;
```



```
SQL> CREATE TABLE migrated_data (
  2      string_date VARCHAR2(20)
  3  );

Table created.

SQL> INSERT INTO migrated_data VALUES ('29/01/2025');

1 row created.

SQL> COMMIT;

Commit complete.

SQL> SELECT TO_DATE(string_date, 'DD/MM/YYYY') AS formatted_date FROM migrated_data;

FORMATTED
---------
29-JAN-25
```

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:

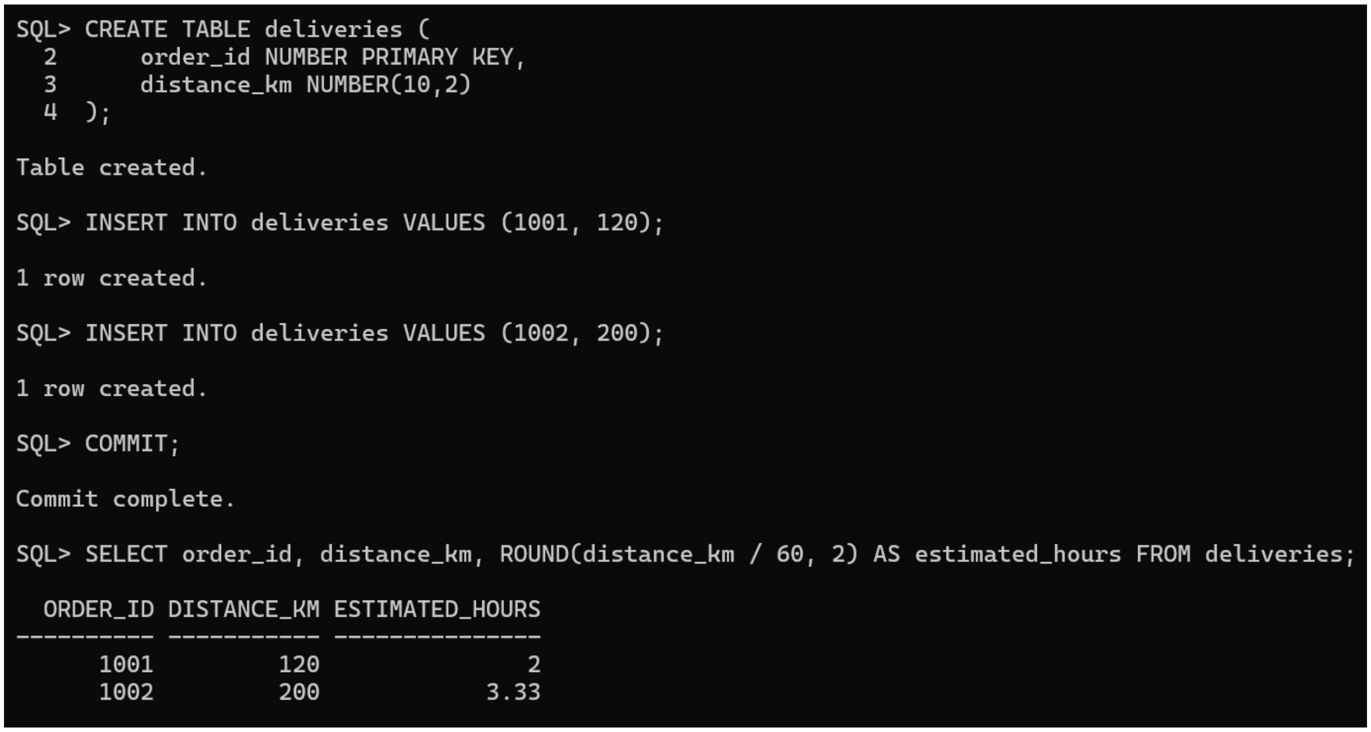
formatted_date
2025-01-29

6 Logistics & Delivery: Calculating Expected Delivery Ti Based on Distance

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;
```



```
SQL> CREATE TABLE deliveries (
  2      order_id NUMBER PRIMARY KEY,
  3      distance_km NUMBER(10,2)
  4  );

Table created.

SQL> INSERT INTO deliveries VALUES (1001, 120);

1 row created.

SQL> INSERT INTO deliveries VALUES (1002, 200);

1 row created.

SQL> COMMIT;

Commit complete.

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

  ORDER_ID DISTANCE_KM ESTIMATED_HOURS
---------- ----------- ---------------
      1001         120               2
      1002         200            3.33
```

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:

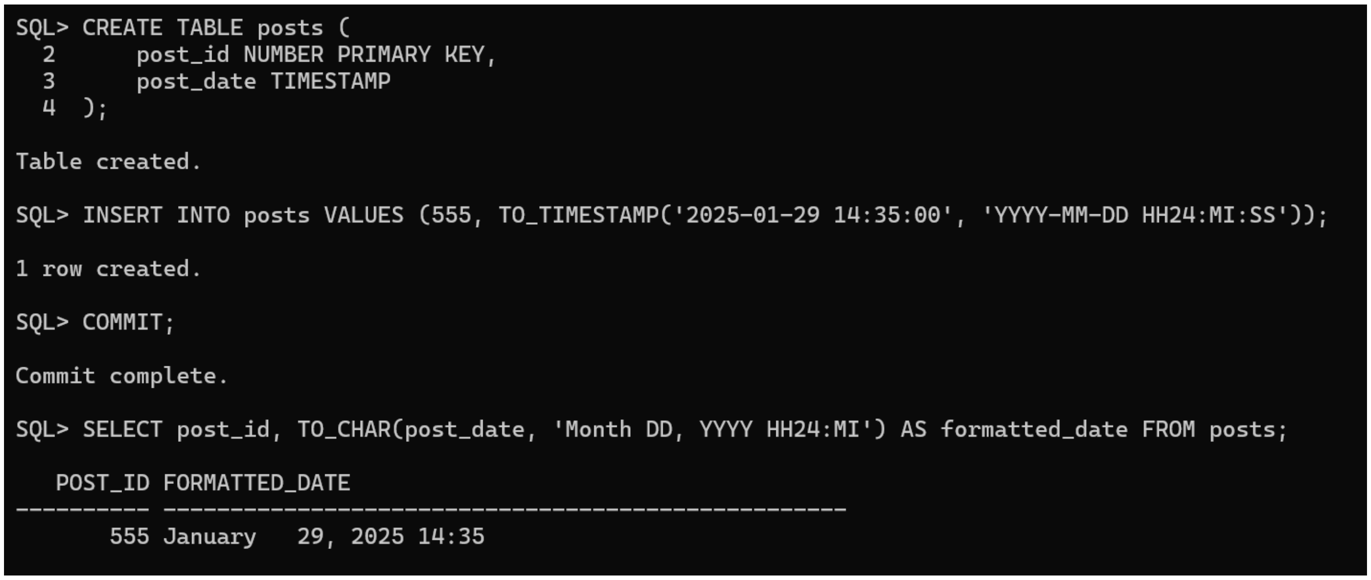
order_id	distance_km	estimated_hours
1001	120	2.00

7️⃣Social Media Analytics: Converting Post Dates in Readable Formats

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;
```



```
SQL> CREATE TABLE posts (
  2      post_id NUMBER PRIMARY KEY,
  3      post_date TIMESTAMP
  4  );

Table created.

SQL> INSERT INTO posts VALUES (555, TO_TIMESTAMP('2025-01-29 14:35:00', 'YYYY-MM-DD HH24:MI:SS'));

1 row created.

SQL> COMMIT;

Commit complete.

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

   POST_ID FORMATTED_DATE
---------- ---------------------------------------------------
       555 January   29, 2025 14:35
```

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')`