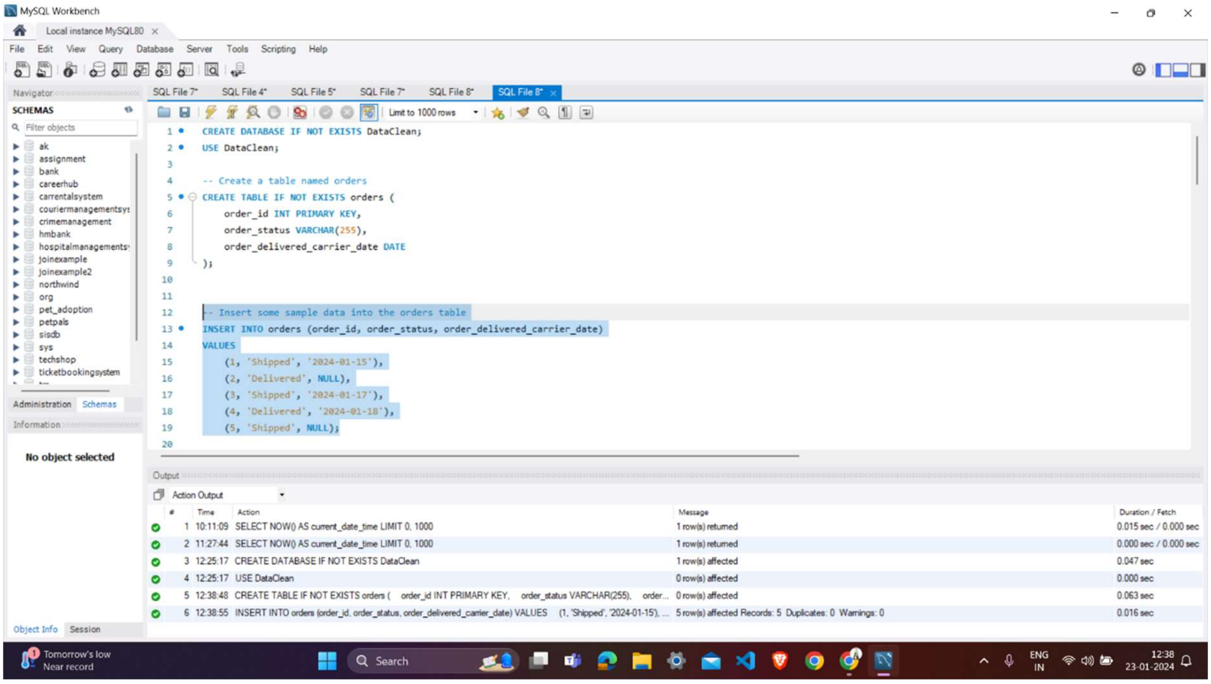
Name: Abhishek Kanoujia

DATA ENGINEERING BATCH 1

DAY 5 ASSIGNMENT

Table of Contents

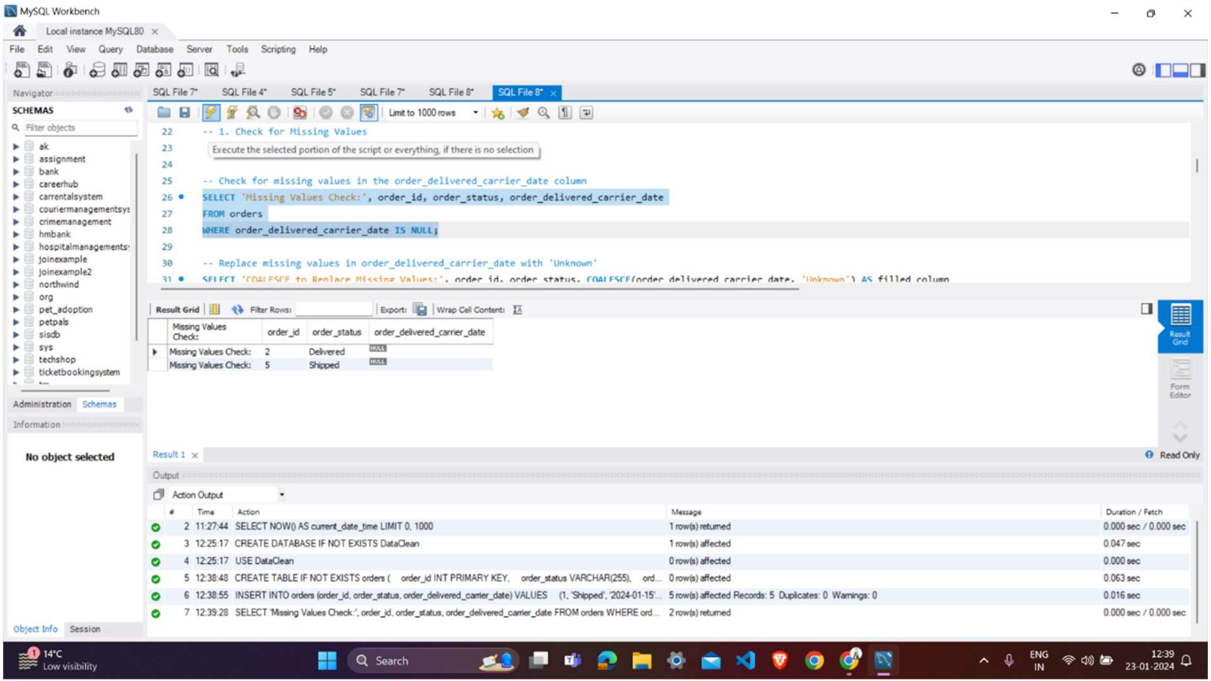
All the queries related to this mentioned in the screen shorts



```sql
CREATE DATABASE IF NOT EXISTS DataClean;
USE DataClean;

-- Create a table named orders
CREATE TABLE IF NOT EXISTS orders (
    order_id INT PRIMARY KEY,
    order_status VARCHAR(255),
    order_delivered_carrier_date DATE
);


-- Insert some sample data into the orders table
INSERT INTO orders (order_id, order_status, order_delivered_carrier_date)
VALUES
    (1, 'Shipped', '2024-01-15'),
    (2, 'Delivered', NULL),
    (3, 'Shipped', '2024-01-17'),
    (4, 'Delivered', '2024-01-18'),
    (5, 'Shipped', NULL);

```

Action Output

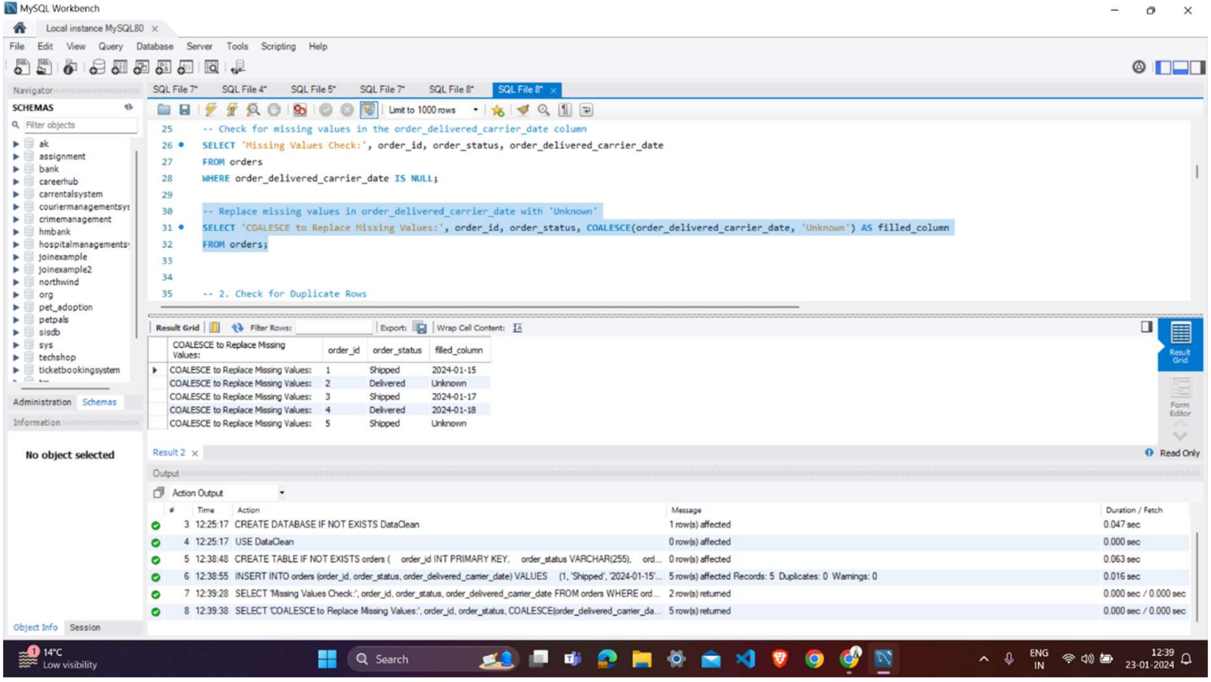
#	Time	Action	Message	Duration / Fetch
1	10:11:09	SELECT NOW() AS current_date_time LIMIT 0, 1000	1 row(s) returned	0.015 sec / 0.000 sec
2	11:27:44	SELECT NOW() AS current_date_time LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec
3	12:25:17	CREATE DATABASE IF NOT EXISTS DataClean	1 row(s) affected	0.047 sec
4	12:25:17	USE DataClean	0 row(s) affected	0.000 sec
5	12:38:48	CREATE TABLE IF NOT EXISTS orders (order_id INT PRIMARY KEY, order_status VARCHAR(255), order...	0 row(s) affected	0.063 sec
6	12:38:55	INSERT INTO orders (order_id, order_status, order_delivered_carrier_date) VALUES (1, 'Shipped', '2024-01-15), ...	5 row(s) affected Records: 5 Duplicates: 0 Warnings: 0	0.016 sec



```sql
-- 1. Check for Missing Values


-- Check for missing values in the order_delivered_carrier_date column
SELECT 'Missing Values Check:', order_id, order_status, order_delivered_carrier_date
FROM orders
WHERE order_delivered_carrier_date IS NULL;

-- Replace missing values in order_delivered_carrier_date with 'Unknown'
SELECT 'COALESCE to Replace Missing Values:', order_id, order_status, COALESCE(order_delivered_carrier_date, 'Unknown') AS filled_column
```

Missing Values Check:	order_id	order_status	order_delivered_carrier_date
Missing Values Check:	2	Delivered	NULL
Missing Values Check:	5	Shipped	NULL

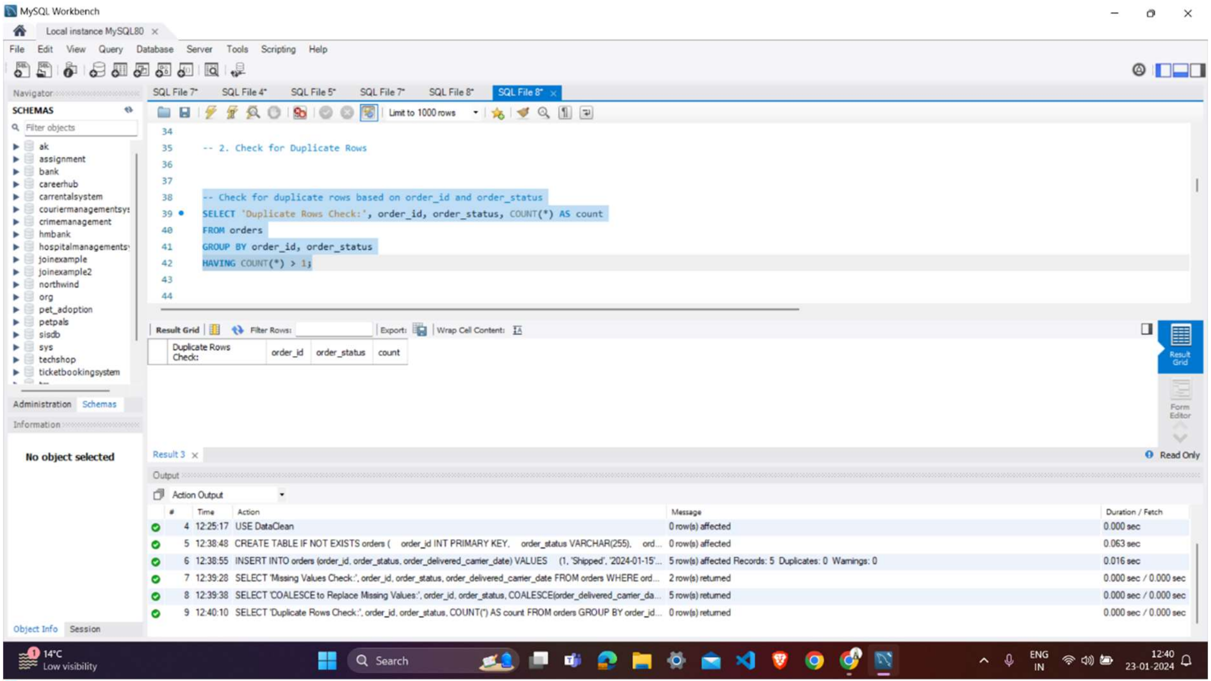
```
#  Time      Action                                                                                                      Message                                                      Duration / Fetch
2  11:27:44  SELECT NOW() AS current_date_time LIMIT 0, 1000                                                             1 row(s) returned                                            0.000 sec / 0.000 sec
3  12:25:17  CREATE DATABASE IF NOT EXISTS DataClean                                                                     1 row(s) affected                                            0.047 sec
4  12:25:17  USE DataClean                                                                                               0 row(s) affected                                            0.000 sec
5  12:38:48  CREATE TABLE IF NOT EXISTS orders (    order_id INT PRIMARY KEY,    order_status VARCHAR(255),    ord...     0 row(s) affected                                            0.063 sec
6  12:38:55  INSERT INTO orders (order_id, order_status, order_delivered_carrier_date) VALUES    (1, 'Shipped', '2024-01-15'...  5 row(s) affected Records: 5  Duplicates: 0  Warnings: 0  0.016 sec
7  12:39:28  SELECT 'Missing Values Check:', order_id, order_status, order_delivered_carrier_date FROM orders WHERE ord...  2 row(s) returned                                            0.000 sec / 0.000 sec
```



```sql
-- Check for missing values in the order_delivered_carrier_date column
SELECT 'Missing Values Check:', order_id, order_status, order_delivered_carrier_date
FROM orders
WHERE order_delivered_carrier_date IS NULL;

-- Replace missing values in order_delivered_carrier_date with 'Unknown'
SELECT 'COALESCE to Replace Missing Values:', order_id, order_status, COALESCE(order_delivered_carrier_date, 'Unknown') AS filled_column
FROM orders;



-- 2. Check for Duplicate Rows
```

COALESCE to Replace Missing Values:	order_id	order_status	filled_column
COALESCE to Replace Missing Values:	1	Shipped	2024-01-15
COALESCE to Replace Missing Values:	2	Delivered	Unknown
COALESCE to Replace Missing Values:	3	Shipped	2024-01-17
COALESCE to Replace Missing Values:	4	Delivered	2024-01-18
COALESCE to Replace Missing Values:	5	Shipped	Unknown

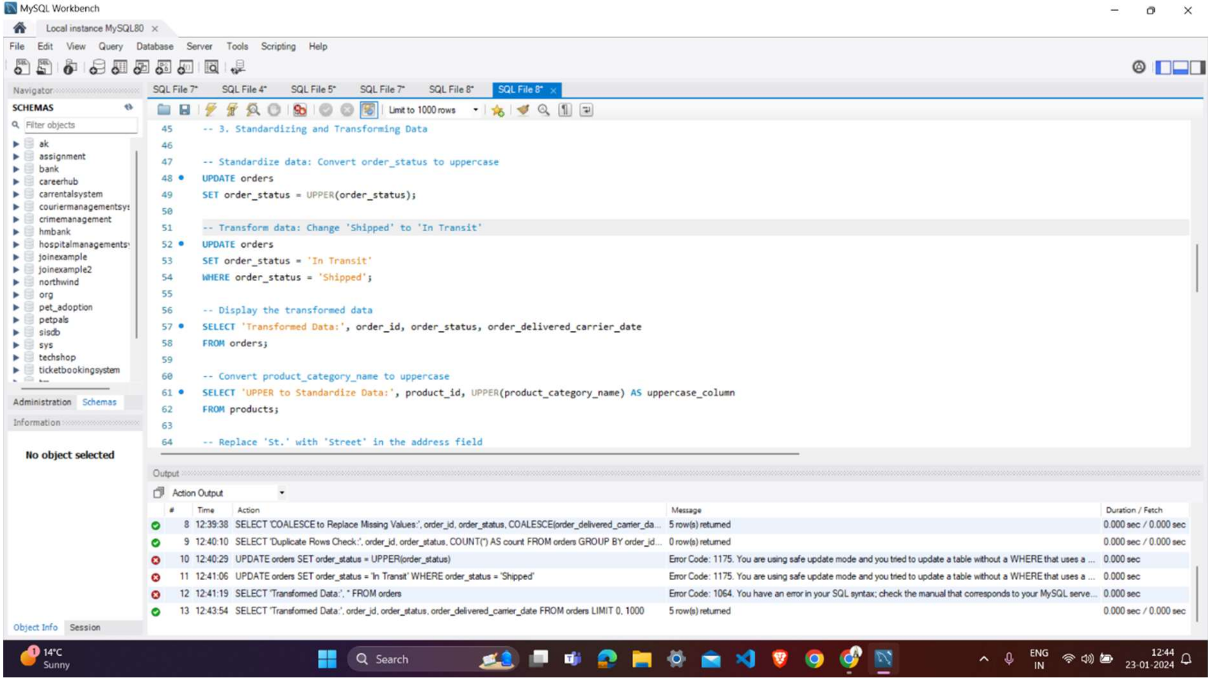
```
#  Time      Action                                                                                                      Message                                                      Duration / Fetch
3  12:25:17  CREATE DATABASE IF NOT EXISTS DataClean                                                                     1 row(s) affected                                            0.047 sec
4  12:25:17  USE DataClean                                                                                               0 row(s) affected                                            0.000 sec
5  12:38:48  CREATE TABLE IF NOT EXISTS orders (    order_id INT PRIMARY KEY,    order_status VARCHAR(255),    ord...     0 row(s) affected                                            0.063 sec
6  12:38:55  INSERT INTO orders (order_id, order_status, order_delivered_carrier_date) VALUES    (1, 'Shipped', '2024-01-15'...  5 row(s) affected Records: 5  Duplicates: 0  Warnings: 0  0.016 sec
7  12:39:28  SELECT 'Missing Values Check:', order_id, order_status, order_delivered_carrier_date FROM orders WHERE ord...  2 row(s) returned                                            0.000 sec / 0.000 sec
8  12:39:38  SELECT 'COALESCE to Replace Missing Values:', order_id, order_status, COALESCE(order_delivered_carrier_da...   5 row(s) returned                                            0.000 sec / 0.000 sec
```



```sql
-- 2. Check for Duplicate Rows


-- Check for duplicate rows based on order_id and order_status
SELECT 'Duplicate Rows Check:', order_id, order_status, COUNT(*) AS count
FROM orders
GROUP BY order_id, order_status
HAVING COUNT(*) > 1;
```

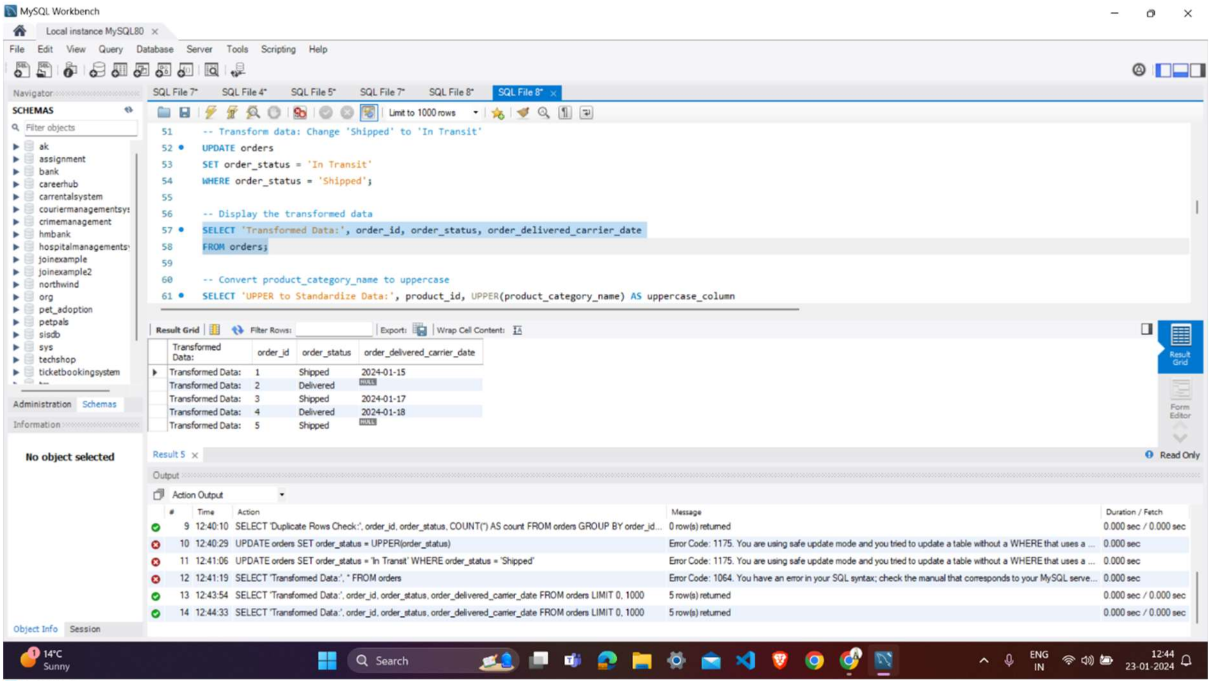
Duplicate Rows Check:	order_id	order_status	count

```
#  Time      Action                                                                                                Message                                                        Duration / Fetch
4  12:25:17  USE DataClean                                                                                         0 row(s) affected                                              0.000 sec
5  12:38:48  CREATE TABLE IF NOT EXISTS orders (    order_id INT PRIMARY KEY,    order_status VARCHAR(255),    ord...   0 row(s) affected                                              0.063 sec
6  12:38:55  INSERT INTO orders (order_id, order_status, order_delivered_carrier_date) VALUES    (1, 'Shipped', '2024-01-15'...   5 row(s) affected Records: 5  Duplicates: 0  Warnings: 0   0.016 sec
7  12:39:28  SELECT 'Missing Values Check:', order_id, order_status, order_delivered_carrier_date FROM orders WHERE ord...   2 row(s) returned                                              0.000 sec / 0.000 sec
8  12:39:38  SELECT 'COALESCE to Replace Missing Values:', order_id, order_status, COALESCE(order_delivered_carrier_da...   5 row(s) returned                                              0.000 sec / 0.000 sec
9  12:40:10  SELECT 'Duplicate Rows Check:', order_id, order_status, COUNT(*) AS count FROM orders GROUP BY order_id...   0 row(s) returned                                              0.000 sec / 0.000 sec
```



```sql
-- 3. Standardizing and Transforming Data

-- Standardize data: Convert order_status to uppercase
UPDATE orders
SET order_status = UPPER(order_status);

-- Transform data: Change 'Shipped' to 'In Transit'
UPDATE orders
SET order_status = 'In Transit'
WHERE order_status = 'Shipped';

-- Display the transformed data
SELECT 'Transformed Data:', order_id, order_status, order_delivered_carrier_date
FROM orders;

-- Convert product_category_name to uppercase
SELECT 'UPPER to Standardize Data:', product_id, UPPER(product_category_name) AS uppercase_column
FROM products;

-- Replace 'St.' with 'Street' in the address field
```

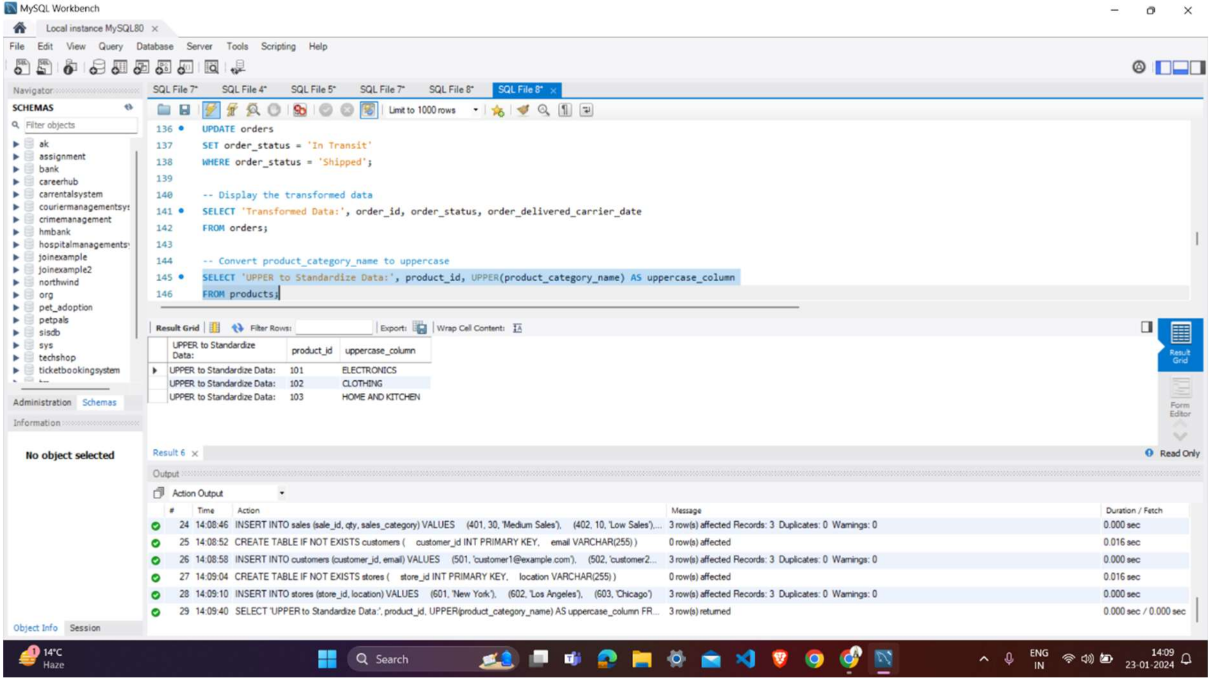
```
#   Time      Action                                                                                                Message                                                                                     Duration / Fetch
8   12:39:38  SELECT 'COALESCE to Replace Missing Values:', order_id, order_status, COALESCE(order_delivered_carrier_da...   5 row(s) returned                                                                           0.000 sec / 0.000 sec
9   12:40:10  SELECT 'Duplicate Rows Check:', order_id, order_status, COUNT(*) AS count FROM orders GROUP BY order_id...   0 row(s) returned                                                                           0.000 sec / 0.000 sec
10  12:40:29  UPDATE orders SET order_status = UPPER(order_status)                                                      Error Code: 1175. You are using safe update mode and you tried to update a table without a WHERE that uses a ...   0.000 sec
11  12:41:06  UPDATE orders SET order_status = 'In Transit' WHERE order_status = 'Shipped'                               Error Code: 1175. You are using safe update mode and you tried to update a table without a WHERE that uses a ...   0.000 sec
12  12:41:19  SELECT 'Transformed Data:', * FROM orders                                                                 Error Code: 1064. You have an error in your SQL syntax; check the manual that corresponds to your MySQL serve...   0.000 sec
13  12:43:54  SELECT 'Transformed Data:', order_id, order_status, order_delivered_carrier_date FROM orders LIMIT 0, 1000    5 row(s) returned                                                                           0.000 sec / 0.000 sec
```



```sql
-- Transform data: Change 'Shipped' to 'In Transit'
UPDATE orders
SET order_status = 'In Transit'
WHERE order_status = 'Shipped';

-- Display the transformed data
SELECT 'Transformed Data:', order_id, order_status, order_delivered_carrier_date
FROM orders;

-- Convert product_category_name to uppercase
SELECT 'UPPER to Standardize Data:', product_id, UPPER(product_category_name) AS uppercase_column
```

Transformed Data:	order_id	order_status	order_delivered_carrier_date
Transformed Data:	1	Shipped	2024-01-15
Transformed Data:	2	Delivered	NULL
Transformed Data:	3	Shipped	2024-01-17
Transformed Data:	4	Delivered	2024-01-18
Transformed Data:	5	Shipped	NULL

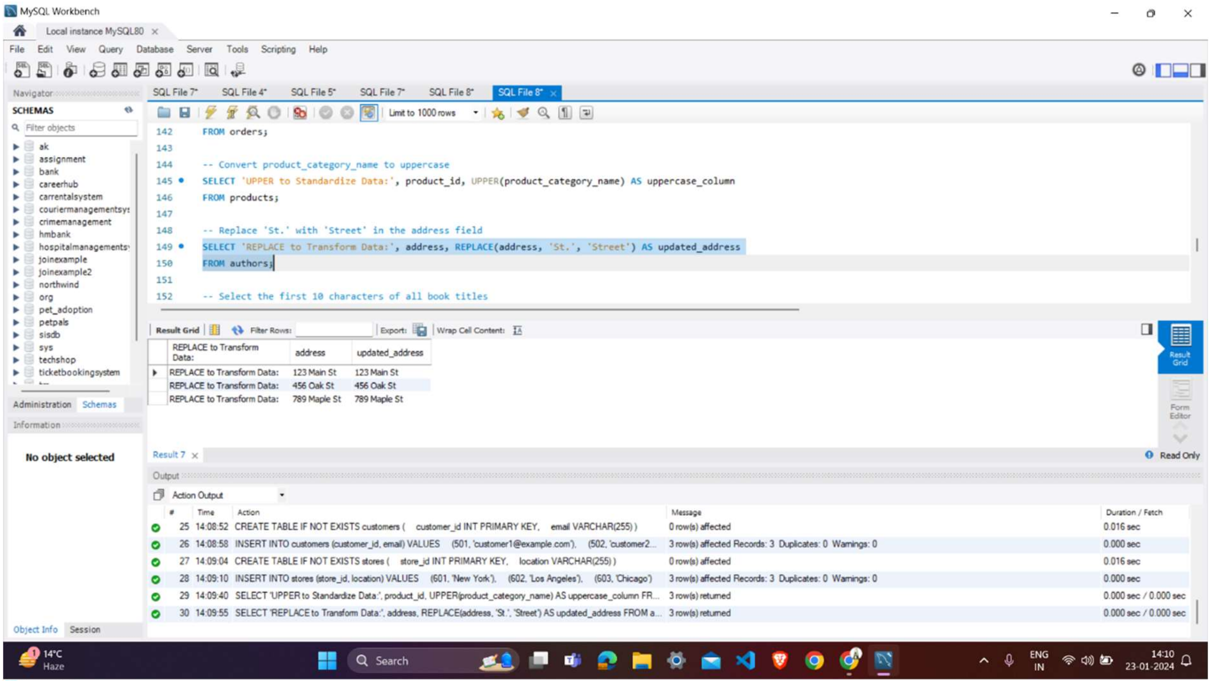
```
#   Time      Action                                                                                          Message                                                                                    Duration / Fetch
9   12:40:10  SELECT 'Duplicate Rows Check:', order_id, order_status, COUNT(*) AS count FROM orders GROUP BY order_id...  0 row(s) returned                                                                          0.000 sec / 0.000 sec
10  12:40:29  UPDATE orders SET order_status = UPPER(order_status)                                            Error Code: 1175. You are using safe update mode and you tried to update a table without a WHERE that uses a ...  0.000 sec
11  12:41:06  UPDATE orders SET order_status = 'In Transit' WHERE order_status = 'Shipped'                     Error Code: 1175. You are using safe update mode and you tried to update a table without a WHERE that uses a ...  0.000 sec
12  12:41:19  SELECT 'Transformed Data:', * FROM orders                                                       Error Code: 1064. You have an error in your SQL syntax; check the manual that corresponds to your MySQL serve...  0.000 sec
13  12:43:54  SELECT 'Transformed Data:', order_id, order_status, order_delivered_carrier_date FROM orders LIMIT 0, 1000  5 row(s) returned                                                            0.000 sec / 0.000 sec
14  12:44:33  SELECT 'Transformed Data:', order_id, order_status, order_delivered_carrier_date FROM orders LIMIT 0, 1000  5 row(s) returned                                                            0.000 sec / 0.000 sec
```



```sql
UPDATE orders
SET order_status = 'In Transit'
WHERE order_status = 'Shipped';

-- Display the transformed data
SELECT 'Transformed Data:', order_id, order_status, order_delivered_carrier_date
FROM orders;

-- Convert product_category_name to uppercase
SELECT 'UPPER to Standardize Data:', product_id, UPPER(product_category_name) AS uppercase_column
FROM products;
```

UPPER to Standardize Data:	product_id	uppercase_column
UPPER to Standardize Data:	101	ELECTRONICS
UPPER to Standardize Data:	102	CLOTHING
UPPER to Standardize Data:	103	HOME AND KITCHEN

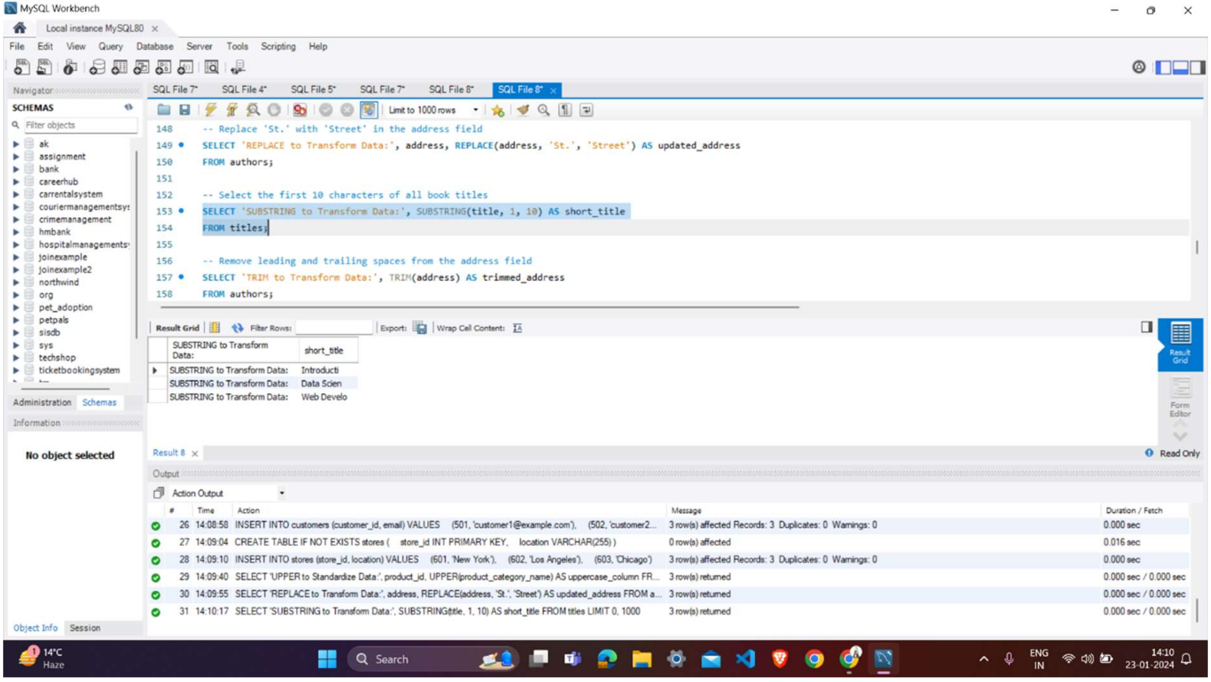
```
#   Time      Action                                                                                          Message                                                     Duration / Fetch
24  14:08:46  INSERT INTO sales (sale_id, qty, sales_category) VALUES    (401, 30, 'Medium Sales'),    (402, 10, 'Low Sales'),...  3 row(s) affected Records: 3  Duplicates: 0  Warnings: 0  0.000 sec
25  14:08:52  CREATE TABLE IF NOT EXISTS customers (    customer_id INT PRIMARY KEY,    email VARCHAR(255) )       0 row(s) affected                                           0.016 sec
26  14:08:58  INSERT INTO customers (customer_id, email) VALUES    (501, 'customer1@example.com'),    (502, 'customer2...  3 row(s) affected Records: 3  Duplicates: 0  Warnings: 0  0.000 sec
27  14:09:04  CREATE TABLE IF NOT EXISTS stores (    store_id INT PRIMARY KEY,    location VARCHAR(255) )          0 row(s) affected                                           0.016 sec
28  14:09:10  INSERT INTO stores (store_id, location) VALUES    (601, 'New York'),    (602, 'Los Angeles'),    (603, 'Chicago')  3 row(s) affected Records: 3  Duplicates: 0  Warnings: 0  0.000 sec
29  14:09:40  SELECT 'UPPER to Standardize Data:', product_id, UPPER(product_category_name) AS uppercase_column FR...  3 row(s) returned                                  0.000 sec / 0.000 sec
```



```sql
FROM orders;

-- Convert product_category_name to uppercase
SELECT 'UPPER to Standardize Data:', product_id, UPPER(product_category_name) AS uppercase_column
FROM products;

-- Replace 'St.' with 'Street' in the address field
SELECT 'REPLACE to Transform Data:', address, REPLACE(address, 'St.', 'Street') AS updated_address
FROM authors;

-- Select the first 10 characters of all book titles
```

REPLACE to Transform Data:	address	updated_address
REPLACE to Transform Data:	123 Main St	123 Main St
REPLACE to Transform Data:	456 Oak St	456 Oak St
REPLACE to Transform Data:	789 Maple St	789 Maple St

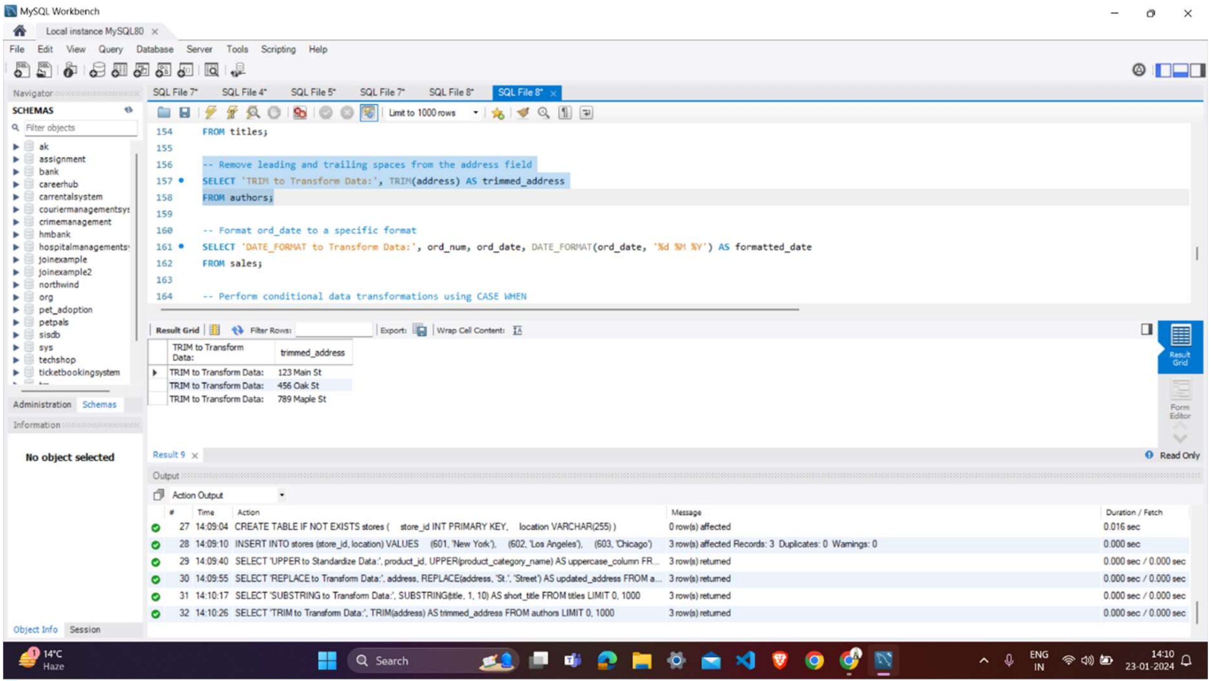
```
#   Time      Action                                                                                              Message                                                      Duration / Fetch
25  14:08:52  CREATE TABLE IF NOT EXISTS customers (    customer_id INT PRIMARY KEY,    email VARCHAR(255) )     0 row(s) affected                                            0.016 sec
26  14:08:58  INSERT INTO customers (customer_id, email) VALUES    (501, 'customer1@example.com'),    (502, 'customer2...   3 row(s) affected Records: 3  Duplicates: 0  Warnings: 0   0.000 sec
27  14:09:04  CREATE TABLE IF NOT EXISTS stores (    store_id INT PRIMARY KEY,    location VARCHAR(255) )          0 row(s) affected                                            0.016 sec
28  14:09:10  INSERT INTO stores (store_id, location) VALUES    (601, 'New York'),    (602, 'Los Angeles'),    (603, 'Chicago')   3 row(s) affected Records: 3  Duplicates: 0  Warnings: 0   0.000 sec
29  14:09:40  SELECT 'UPPER to Standardize Data:', product_id, UPPER(product_category_name) AS uppercase_column FR...   3 row(s) returned                                 0.000 sec / 0.000 sec
30  14:09:55  SELECT 'REPLACE to Transform Data:', address, REPLACE(address, 'St.', 'Street') AS updated_address FROM a...   3 row(s) returned                   0.000 sec / 0.000 sec
```



```sql
-- Replace 'St.' with 'Street' in the address field
SELECT 'REPLACE to Transform Data:', address, REPLACE(address, 'St.', 'Street') AS updated_address
FROM authors;

-- Select the first 10 characters of all book titles
SELECT 'SUBSTRING to Transform Data:', SUBSTRING(title, 1, 10) AS short_title
FROM titles;

-- Remove leading and trailing spaces from the address field
SELECT 'TRIM to Transform Data:', TRIM(address) AS trimmed_address
FROM authors;
```

SUBSTRING to Transform Data:	short_title
SUBSTRING to Transform Data:	Introducti
SUBSTRING to Transform Data:	Data Scien
SUBSTRING to Transform Data:	Web Develo

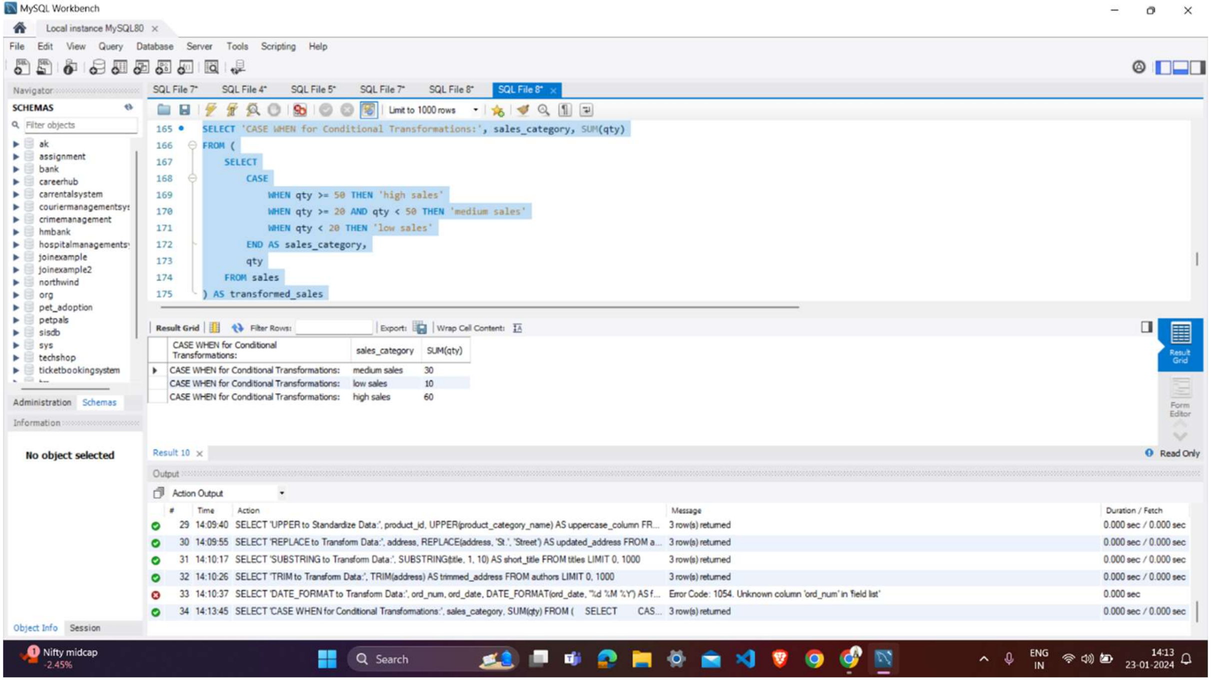
```
#   Time      Action                                                                                              Message                                                      Duration / Fetch
26  14:08:58  INSERT INTO customers (customer_id, email) VALUES    (501, 'customer1@example.com'),    (502, 'customer2...   3 row(s) affected Records: 3  Duplicates: 0  Warnings: 0   0.000 sec
27  14:09:04  CREATE TABLE IF NOT EXISTS stores (    store_id INT PRIMARY KEY,    location VARCHAR(255) )          0 row(s) affected                                            0.016 sec
28  14:09:10  INSERT INTO stores (store_id, location) VALUES    (601, 'New York'),    (602, 'Los Angeles'),    (603, 'Chicago')   3 row(s) affected Records: 3  Duplicates: 0  Warnings: 0   0.000 sec
29  14:09:40  SELECT 'UPPER to Standardize Data:', product_id, UPPER(product_category_name) AS uppercase_column FR...   3 row(s) returned                                 0.000 sec / 0.000 sec
30  14:09:55  SELECT 'REPLACE to Transform Data:', address, REPLACE(address, 'St.', 'Street') AS updated_address FROM a...   3 row(s) returned                   0.000 sec / 0.000 sec
31  14:10:17  SELECT 'SUBSTRING to Transform Data:', SUBSTRING(title, 1, 10) AS short_title FROM titles LIMIT 0, 1000   3 row(s) returned                              0.000 sec / 0.000 sec
```



```sql
FROM titles;

-- Remove leading and trailing spaces from the address field
SELECT 'TRIM to Transform Data:', TRIM(address) AS trimmed_address
FROM authors;

-- Format ord_date to a specific format
SELECT 'DATE_FORMAT to Transform Data:', ord_num, ord_date, DATE_FORMAT(ord_date, '%d %M %Y') AS formatted_date
FROM sales;

-- Perform conditional data transformations using CASE WHEN
```

TRIM to Transform Data:	trimmed_address
TRIM to Transform Data:	123 Main St
TRIM to Transform Data:	456 Oak St
TRIM to Transform Data:	789 Maple St

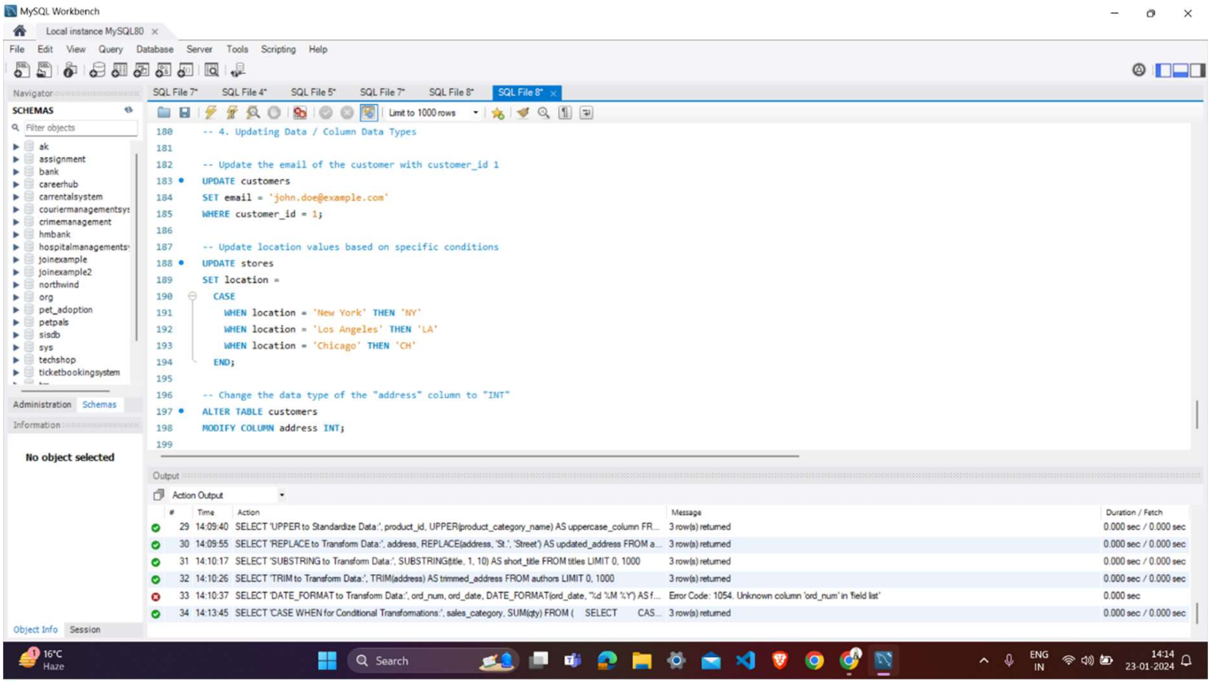
```
#   Time      Action                                                                                          Message                                                        Duration / Fetch
27  14:09:04  CREATE TABLE IF NOT EXISTS stores (    store_id INT PRIMARY KEY,    location VARCHAR(255) )      0 row(s) affected                                              0.016 sec
28  14:09:10  INSERT INTO stores (store_id, location) VALUES    (601, 'New York'),    (602, 'Los Angeles'),    (603, 'Chicago')   3 row(s) affected Records: 3  Duplicates: 0  Warnings: 0   0.000 sec
29  14:09:40  SELECT 'UPPER to Standardize Data:', product_id, UPPER(product_category_name) AS uppercase_column FR...   3 row(s) returned   0.000 sec / 0.000 sec
30  14:09:55  SELECT 'REPLACE to Transform Data:', address, REPLACE(address, 'St.', 'Street') AS updated_address FROM a...   3 row(s) returned   0.000 sec / 0.000 sec
31  14:10:17  SELECT 'SUBSTRING to Transform Data:', SUBSTRING(title, 1, 10) AS short_title FROM titles LIMIT 0, 1000   3 row(s) returned   0.000 sec / 0.000 sec
32  14:10:26  SELECT 'TRIM to Transform Data:', TRIM(address) AS trimmed_address FROM authors LIMIT 0, 1000   3 row(s) returned   0.000 sec / 0.000 sec
```



```sql
SELECT 'CASE WHEN for Conditional Transformations:', sales_category, SUM(qty)
FROM (
    SELECT
        CASE
            WHEN qty >= 50 THEN 'high sales'
            WHEN qty >= 20 AND qty < 50 THEN 'medium sales'
            WHEN qty < 20 THEN 'low sales'
        END AS sales_category,
        qty
    FROM sales
) AS transformed_sales
```

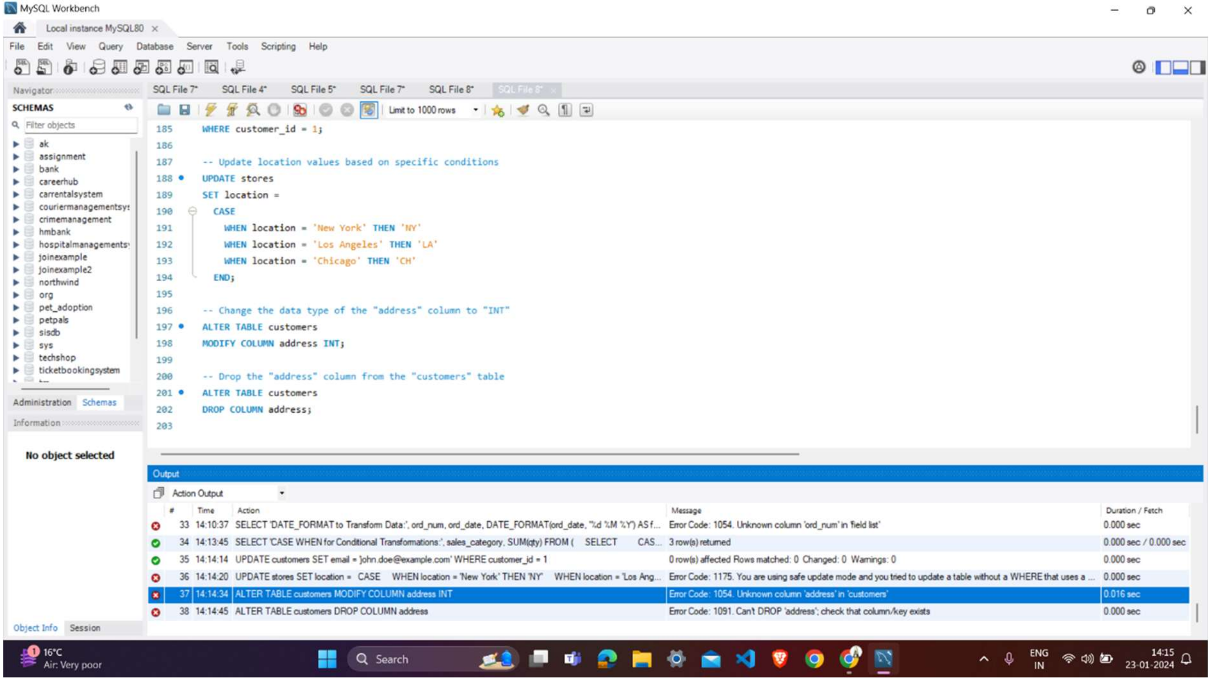
CASE WHEN for Conditional Transformations:	sales_category	SUM(qty)
CASE WHEN for Conditional Transformations:	medium sales	30
CASE WHEN for Conditional Transformations:	low sales	10
CASE WHEN for Conditional Transformations:	high sales	60

```
#   Time      Action                                                                                          Message                                                        Duration / Fetch
29  14:09:40  SELECT 'UPPER to Standardize Data:', product_id, UPPER(product_category_name) AS uppercase_column FR...   3 row(s) returned   0.000 sec / 0.000 sec
30  14:09:55  SELECT 'REPLACE to Transform Data:', address, REPLACE(address, 'St.', 'Street') AS updated_address FROM a...   3 row(s) returned   0.000 sec / 0.000 sec
31  14:10:17  SELECT 'SUBSTRING to Transform Data:', SUBSTRING(title, 1, 10) AS short_title FROM titles LIMIT 0, 1000   3 row(s) returned   0.000 sec / 0.000 sec
32  14:10:26  SELECT 'TRIM to Transform Data:', TRIM(address) AS trimmed_address FROM authors LIMIT 0, 1000   3 row(s) returned   0.000 sec / 0.000 sec
33  14:10:37  SELECT 'DATE_FORMAT to Transform Data:', ord_num, ord_date, DATE_FORMAT(ord_date, '%d %M %Y') AS f...   Error Code: 1054. Unknown column 'ord_num' in 'field list'   0.000 sec
34  14:13:45  SELECT 'CASE WHEN for Conditional Transformations:', sales_category, SUM(qty) FROM (    SELECT       CAS...   3 row(s) returned   0.000 sec / 0.000 sec
```



```sql
-- 4. Updating Data / Column Data Types

-- Update the email of the customer with customer_id 1
UPDATE customers
SET email = 'john.doe@example.com'
WHERE customer_id = 1;

-- Update location values based on specific conditions
UPDATE stores
SET location =
  CASE
    WHEN location = 'New York' THEN 'NY'
    WHEN location = 'Los Angeles' THEN 'LA'
    WHEN location = 'Chicago' THEN 'CH'
  END;

-- Change the data type of the "address" column to "INT"
ALTER TABLE customers
MODIFY COLUMN address INT;

```

	#	Time	Action	Message	Duration / Fetch
✓	29	14:09:40	SELECT 'UPPER to Standardize Data:', product_id, UPPER(product_category_name) AS uppercase_column FR...	3 row(s) returned	0.000 sec / 0.000 sec
✓	30	14:09:55	SELECT 'REPLACE to Transform Data:', address, REPLACE(address, 'St.', 'Street') AS updated_address FROM a...	3 row(s) returned	0.000 sec / 0.000 sec
✓	31	14:10:17	SELECT 'SUBSTRING to Transform Data:', SUBSTRING(title, 1, 10) AS short_title FROM titles LIMIT 0, 1000	3 row(s) returned	0.000 sec / 0.000 sec
✓	32	14:10:26	SELECT 'TRIM to Transform Data:', TRIM(address) AS trimmed_address FROM authors LIMIT 0, 1000	3 row(s) returned	0.000 sec / 0.000 sec
✗	33	14:10:37	SELECT 'DATE_FORMAT to Transform Data:', ord_num, ord_date, DATE_FORMAT(ord_date, "%d %M %Y") AS f...	Error Code: 1054. Unknown column 'ord_num' in 'field list'	0.000 sec
✓	34	14:13:45	SELECT 'CASE WHEN for Conditional Transformations:', sales_category, SUM(qty) FROM (SELECT CAS...	3 row(s) returned	0.000 sec / 0.000 sec



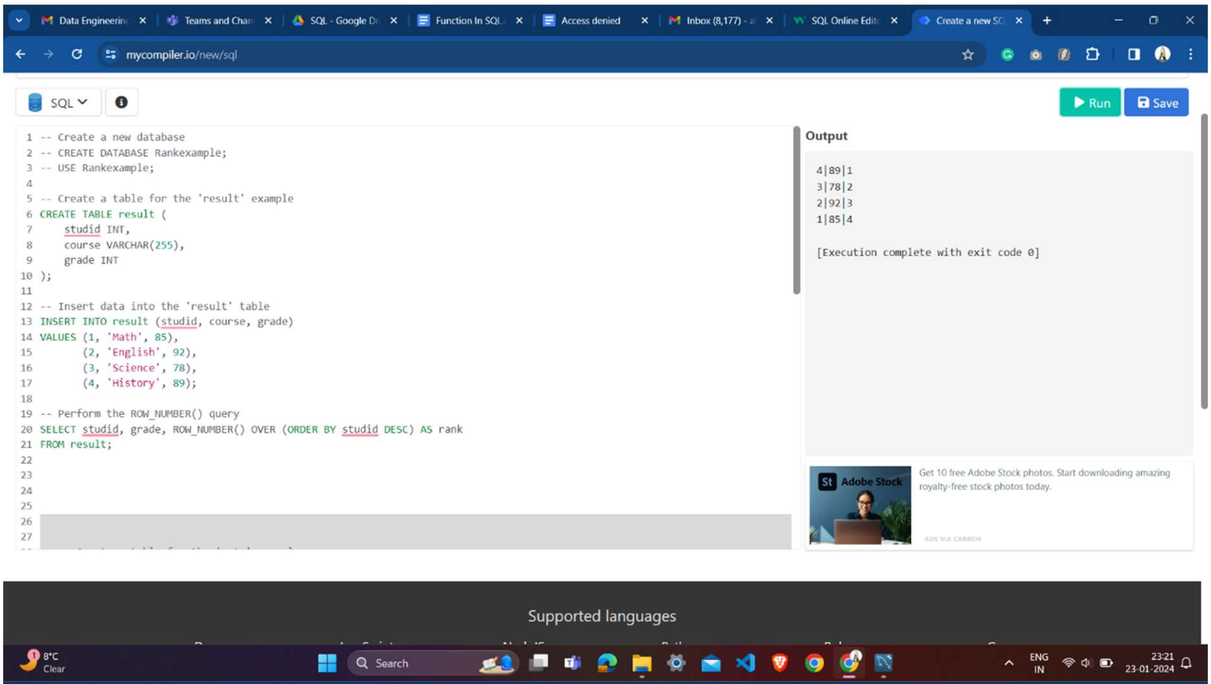
```sql
WHERE customer_id = 1;

-- Update location values based on specific conditions
UPDATE stores
SET location =
  CASE
    WHEN location = 'New York' THEN 'NY'
    WHEN location = 'Los Angeles' THEN 'LA'
    WHEN location = 'Chicago' THEN 'CH'
  END;

-- Change the data type of the "address" column to "INT"
ALTER TABLE customers
MODIFY COLUMN address INT;

-- Drop the "address" column from the "customers" table
ALTER TABLE customers
DROP COLUMN address;

```

	#	Time	Action	Message	Duration / Fetch
✗	33	14:10:37	SELECT 'DATE_FORMAT to Transform Data:', ord_num, ord_date, DATE_FORMAT(ord_date, "%d %M %Y") AS f...	Error Code: 1054. Unknown column 'ord_num' in 'field list'	0.000 sec
✓	34	14:13:45	SELECT 'CASE WHEN for Conditional Transformations:', sales_category, SUM(qty) FROM (SELECT CAS...	3 row(s) returned	0.000 sec / 0.000 sec
✓	35	14:14:14	UPDATE customers SET email = 'john.doe@example.com' WHERE customer_id = 1	0 row(s) affected Rows matched: 0 Changed: 0 Warnings: 0	0.000 sec
✗	36	14:14:20	UPDATE stores SET location = CASE WHEN location = 'New York' THEN 'NY' WHEN location = 'Los Ang...	Error Code: 1175. You are using safe update mode and you tried to update a table without a WHERE that uses a ...	0.000 sec
✗	37	14:14:34	ALTER TABLE customers MODIFY COLUMN address INT	Error Code: 1054. Unknown column 'address' in 'customers'	0.016 sec
✗	38	14:14:45	ALTER TABLE customers DROP COLUMN address	Error Code: 1091. Can't DROP 'address'; check that column/key exists	0.000 sec

Using Rank Functions

The functions are used to generate sequence numbers for each row for providing a rank based on the specified condition. The functions are provided to rank the records in the table. The following functions are provided for the rank records.

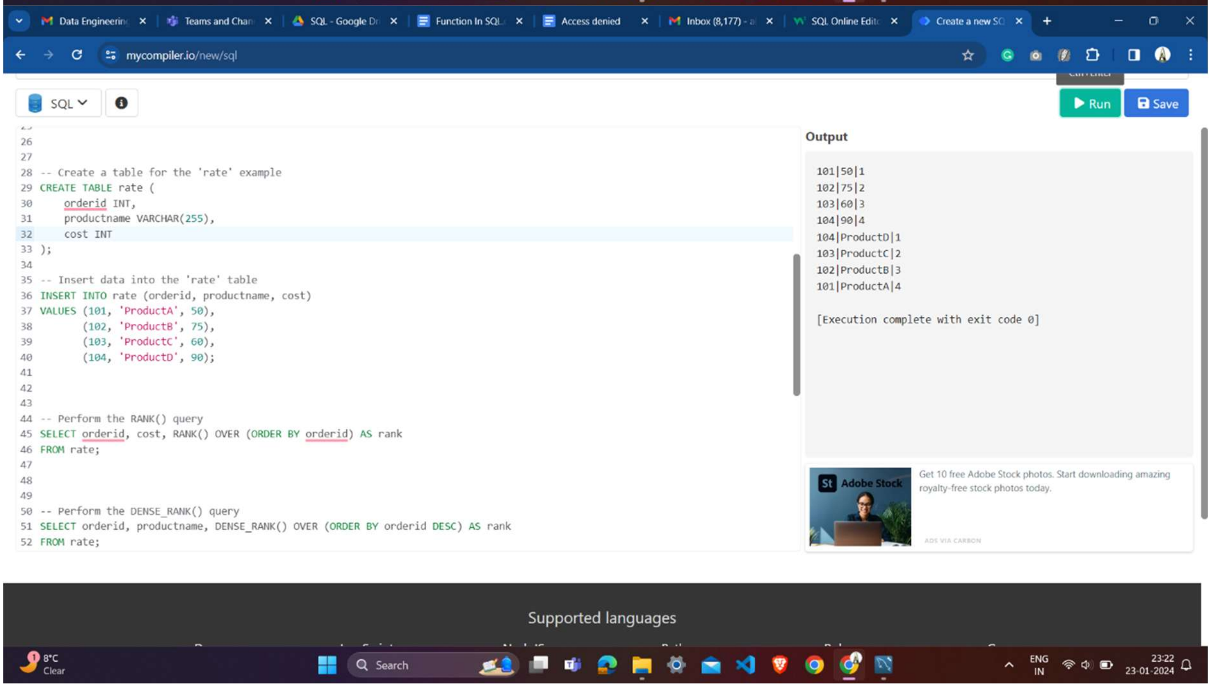
1) row_number()
2) rank ()
3) dense_rank ()
4) ntile ()



```sql
-- Create a new database
-- CREATE DATABASE Rankexample;
-- USE Rankexample;

-- Create a table for the 'result' example
CREATE TABLE result (
    studid INT,
    course VARCHAR(255),
    grade INT
);

-- Insert data into the 'result' table
INSERT INTO result (studid, course, grade)
VALUES (1, 'Math', 85),
       (2, 'English', 92),
       (3, 'Science', 78),
       (4, 'History', 89);

-- Perform the ROW_NUMBER() query
SELECT studid, grade, ROW_NUMBER() OVER (ORDER BY studid DESC) AS rank
FROM result;
```

Output

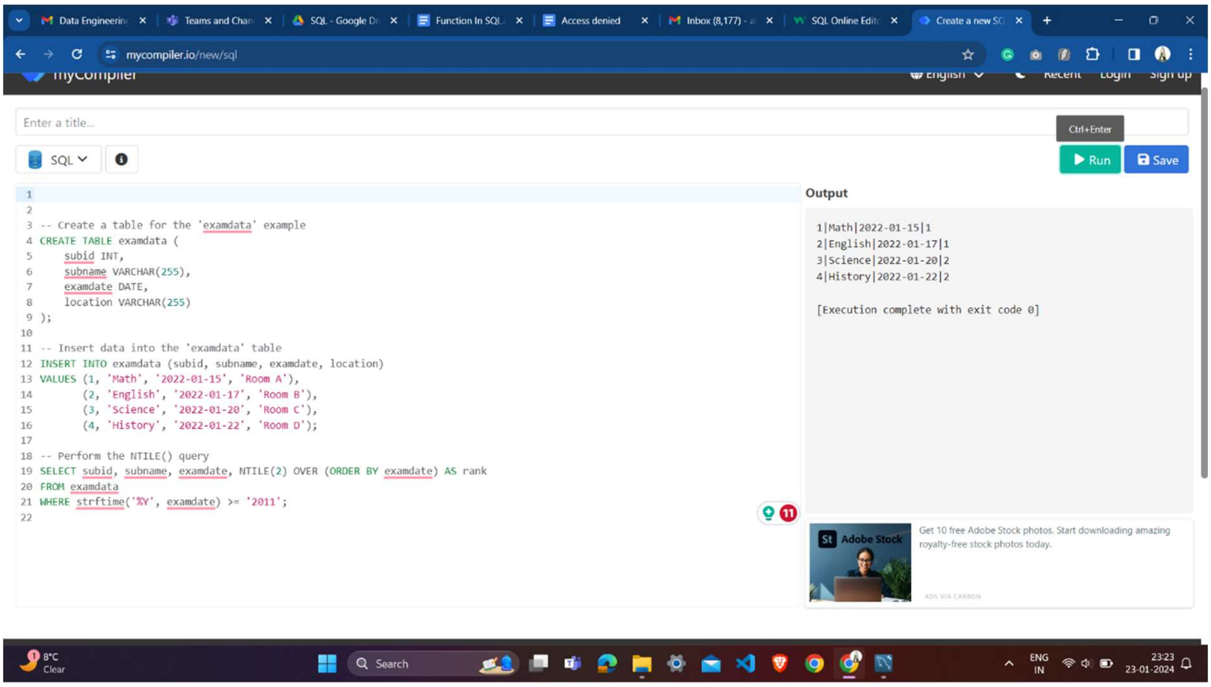
```
4|89|1
3|78|2
2|92|3
1|85|4

[Execution complete with exit code 0]
```



```sql


-- Create a table for the 'rate' example
CREATE TABLE rate (
    orderid INT,
    productname VARCHAR(255),
    cost INT
);

-- Insert data into the 'rate' table
INSERT INTO rate (orderid, productname, cost)
VALUES (101, 'ProductA', 50),
       (102, 'ProductB', 75),
       (103, 'ProductC', 60),
       (104, 'ProductD', 90);



-- Perform the RANK() query
SELECT orderid, cost, RANK() OVER (ORDER BY orderid) AS rank
FROM rate;



-- Perform the DENSE_RANK() query
SELECT orderid, productname, DENSE_RANK() OVER (ORDER BY orderid DESC) AS rank
FROM rate;
```

Output

```
101|50|1
102|75|2
103|60|3
104|90|4
104|ProductD|1
103|ProductC|2
102|ProductB|3
101|ProductA|4

[Execution complete with exit code 0]
```



```sql


-- Create a table for the 'examdata' example
CREATE TABLE examdata (
    subid INT,
    subname VARCHAR(255),
    examdate DATE,
    location VARCHAR(255)
);

-- Insert data into the 'examdata' table
INSERT INTO examdata (subid, subname, examdate, location)
VALUES (1, 'Math', '2022-01-15', 'Room A'),
       (2, 'English', '2022-01-17', 'Room B'),
       (3, 'Science', '2022-01-20', 'Room C'),
       (4, 'History', '2022-01-22', 'Room D');

-- Perform the NTILE() query
SELECT subid, subname, examdate, NTILE(2) OVER (ORDER BY examdate) AS rank
FROM examdata
WHERE strftime('%Y', examdate) >= '2011';

```

```
1|Math|2022-01-15|1
2|English|2022-01-17|1
3|Science|2022-01-20|2
4|History|2022-01-22|2

[Execution complete with exit code 0]
```

SQL Stored Procedures for SQL Server

A stored procedure is a prepared SQL code that you can save, so the code can be reused over and over again.

So if you have an SQL query that you write over and over again, save it as a stored procedure, and then just call it to execute it.

You can also pass parameters to a stored procedure, so that the stored procedure can act based on the parameter value(s) that is passed.

```
-- -- Create Database
CREATE DATABASE YourDatabaseName;
USE YourDatabaseName;
```

```
-- Create Table
CREATE TABLE Customers (
    CustomerID INT PRIMARY KEY,
    CustomerName NVARCHAR(255),
    ContactName NVARCHAR(255),
    Address NVARCHAR(255),
    City NVARCHAR(255),
    PostalCode NVARCHAR(20),
    Country NVARCHAR(255)
);
```

```
-- Insert Values
INSERT INTO Customers (CustomerID, CustomerName, ContactName, Address, City, PostalCode, Country)
VALUES
```

```
(1, 'Alfreds Futterkiste', 'Maria Anders', 'Obere Str. 57', 'Berlin', '12209', 'Germany'),
(2, 'Ana Trujillo Emparedados y helados', 'Ana Trujillo', 'Avda. de la Constitución 2222', 'México D.F.', '05021', 'Mexico'),
(3, 'Antonio Moreno Taquería', 'Antonio Moreno', 'Mataderos 2312', 'México D.F.', '05023', 'Mexico'),
(4, 'Around the Horn', 'Thomas Hardy', '120 Hanover Sq.', 'London', 'WA1 1DP', 'UK'),
(5, 'Berglunds snabbköp', 'Christina Berglund', 'Berguvsvägen 8', 'Luleå', 'S-958 22', 'Sweden');


-- Stored Procedure: Select All Customers
CREATE PROCEDURE SelectAllCustomers
AS
BEGIN
   SELECT * FROM Customers;
END;


-- Stored Procedure: Select Customers by City
CREATE PROCEDURE SelectCustomersByCity
   @City NVARCHAR(255)
AS
BEGIN
   SELECT * FROM Customers WHERE City = @City;
END;


-- Stored Procedure: Select Customers by City and Postal Code
CREATE PROCEDURE SelectCustomersByCityAndPostalCode
   @City NVARCHAR(255),
    @PostalCode NVARCHAR(20)
AS
BEGIN
    SELECT * FROM Customers WHERE City = @City AND PostalCode = @PostalCode;
END;
```

```
-- Execute the stored procedures
EXEC SelectAllCustomers;
EXEC SelectCustomersByCity @City = 'London';
EXEC SelectCustomersByCityAndPostalCode @City = 'London', @PostalCode = 'WA1 1DP';
```