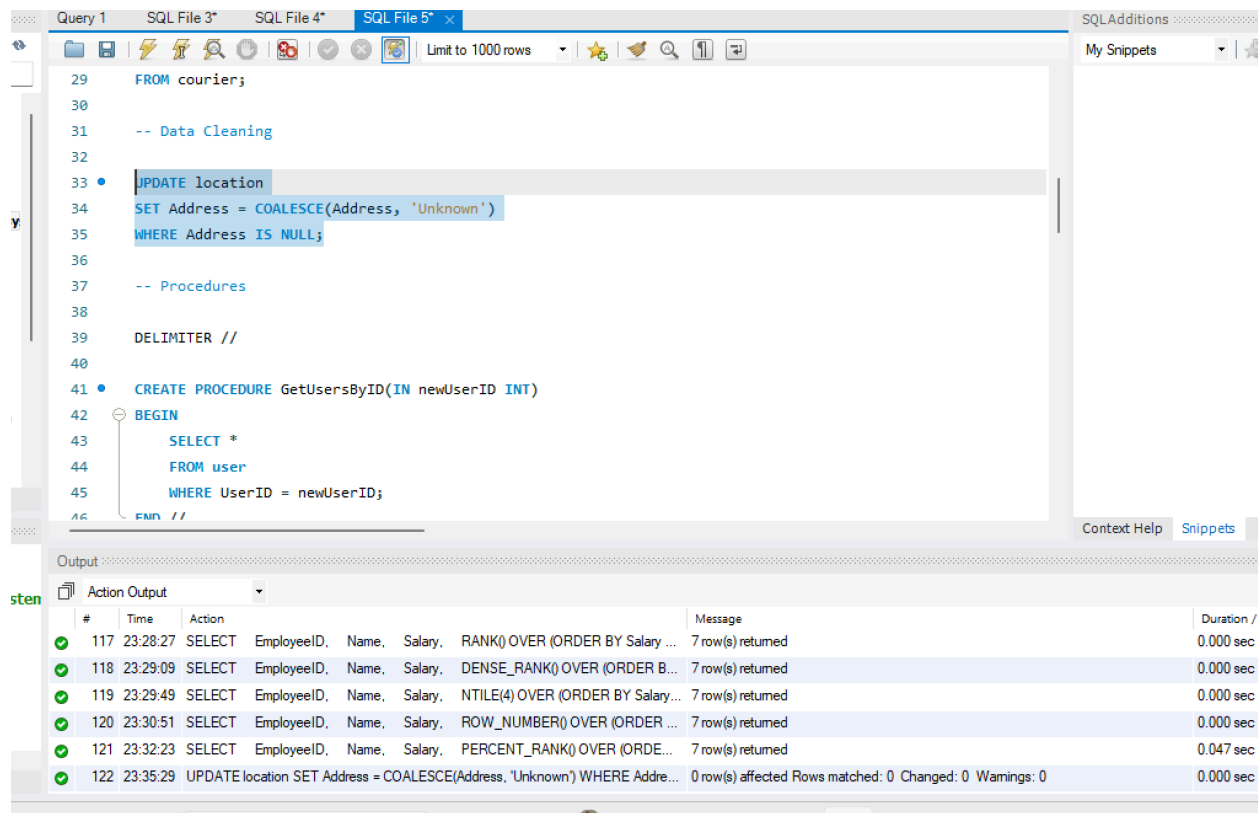


Asmita Porwal
Batch-1
Day-5
23/1/2023
Data engineering

Data Cleaning And Transformation :

Check for missing value

IsNull and coalesce



The screenshot displays the SQL Server Enterprise Manager interface. The top pane shows a query window with the following SQL code:

```
29 FROM courier;  
30  
31 -- Data Cleaning  
32  
33 • UPDATE location  
34 SET Address = COALESCE(Address, 'Unknown')  
35 WHERE Address IS NULL;  
36  
37 -- Procedures  
38  
39 DELIMITER //  
40  
41 • CREATE PROCEDURE GetUsersByID(IN newUserID INT)  
42 BEGIN  
43 SELECT *  
44 FROM user  
45 WHERE UserID = newUserID;  
46 END //
```

The bottom pane shows the 'Output' window with the 'Action Output' tab selected. It displays a table of execution results:

#	Time	Action	Message	Duration /
✓ 117	23:28:27	SELECT EmployeeID, Name, Salary, RANK() OVER (ORDER BY Salary ...	7 row(s) returned	0.000 sec
✓ 118	23:29:09	SELECT EmployeeID, Name, Salary, DENSE_RANK() OVER (ORDER B...	7 row(s) returned	0.000 sec
✓ 119	23:29:49	SELECT EmployeeID, Name, Salary, NTILE(4) OVER (ORDER BY Salary...	7 row(s) returned	0.000 sec
✓ 120	23:30:51	SELECT EmployeeID, Name, Salary, ROW_NUMBER() OVER (ORDER ...	7 row(s) returned	0.000 sec
✓ 121	23:32:23	SELECT EmployeeID, Name, Salary, PERCENT_RANK() OVER (ORDE...	7 row(s) returned	0.047 sec
✓ 122	23:35:29	UPDATE location SET Address = COALESCE(Address, 'Unknown') WHERE Addre...	0 row(s) affected Rows matched: 0 Changed: 0 Warnings: 0	0.000 sec

Check for duplicates:

Group by and having: find duplicate records

Limit to 1000 rows

```

11 • select * from location;
12
13 -- Check for duplicates:
14 • SELECT
15     LocationID,
16     count(CourierID) as no_of_courier
17 FROM payment
18 GROUP BY LocationID having no_of_courier>1;
19
20 -- Data Formatting

```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: [IA](#)

	LocationID	no_of_courier
▶	8	2

Distinct and delete from:

Query 1 | SQL File 3* | SQL File 4* | SQL File 5* x

Limit to 1000 rows

```

18 GROUP BY LocationID having no_of_courier>1;
19
20 • select distinct CourierID from payment;
21 • delete from payment where PaymentID =8;
22 • select * from payment;
23 -- Data Formatting
24
25 • SELECT CourierID, DATE_FORMAT(DeliveryDate, '%d-%m-%y') AS formatted_delivery_date
26 FROM courier;
27

```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: [I](#)

	PaymentID	CourierID	LocationID	Amount	PaymentDate	EmployeeID
	4	4	2	30.00	NULL	NULL
	5	5	5	15.00	2023-12-12	NULL
	6	6	8	50.00	2023-12-15	NULL
	7	7	6	75.00	2023-12-10	NULL
*	NULL	NULL	NULL	NULL	NULL	NULL

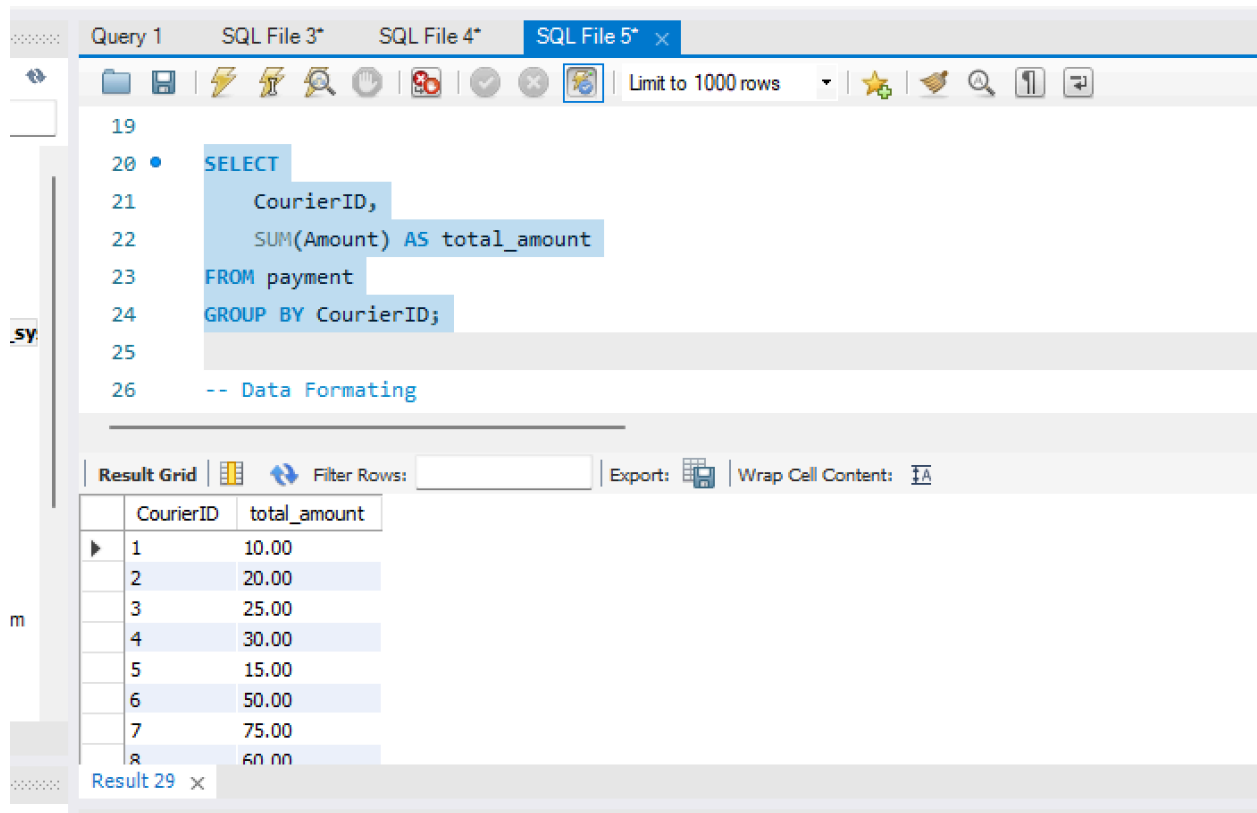
payment 51 x

Output:

Standardizing and transforming data

For consistency and accurate analysis

Aggregate Function :



The screenshot shows a SQL IDE interface with a query editor and a results grid. The query editor contains the following SQL code:

```
19
20 • SELECT
21     CourierID,
22     SUM(Amount) AS total_amount
23 FROM payment
24 GROUP BY CourierID;
25
26 -- Data Formatting
```

The results grid displays the output of the query, showing two columns: CourierID and total_amount. The data is as follows:

CourierID	total_amount
1	10.00
2	20.00
3	25.00
4	30.00
5	15.00
6	50.00
7	75.00
8	60.00

Data formatting :

Limit to 1000 rows

```

25
26      -- Data Formatting
27
28 • SELECT CourierID, DATE_FORMAT(DeliveryDate, '%d-%m-%y') AS formatted_delivery_date
29 FROM courier;
30
31      -- Data Cleaning
32

```

Result Grid

	CourierID	formatted_delivery_date
1	1	15-12-23
2	2	10-12-23
3	3	18-12-23
4	4	NULL
5	5	12-12-23
6	6	15-12-23
7	7	10-12-23
8	8	18-12-23

Result 36 x

Read Only

Procedures :

```

39 WHERE Address IS NULL;
40
41      -- Procedures
42
43 DELIMITER //
44
45 • CREATE PROCEDURE GetUsersByID(IN newUserID INT)
46 BEGIN
47     SELECT *
48     FROM user
49     WHERE UserID = newUserID;
50 END //
51
52 DELIMITER ;
53
54

```

```
54
55 DELIMITER //
56
57 • CREATE PROCEDURE CalculateTotalPayment(IN p_CourierID INT)
58 BEGIN
59     DECLARE totalAmount DECIMAL(10, 2);
60
61     SELECT SUM(Amount) INTO totalAmount
62     FROM payment
63     WHERE CourierID = p_CourierID;
64
65     SELECT totalAmount AS TotalPaymentAmount;
66 END //
67
68 DELIMITER ;
```



79

80 • -- Calling Procedures

81

82 CALL GetUsersByID(2);

83

Result Grid					
Filter Rows: <input type="text"/>					
Export:  Wrap Cell Content: 					
UserID	Name	Email	Password	ContactNumber	Address
2	Jane Smith	jane@example.com	securepass	987-654-3210	456 Elm St

83

84 • `CALL CalculateTotalPayment(1);`

85

86 • `CALL UpdateCourierStatus(3, 'In Transit');`

Result Grid |  Filter Rows: | Export:  | Wrap Cell Content: 

TotalPaymentAmount
10.00

Rank Function:

Rank()

89

90 • `SELECT`

91 `EmployeeID,`

92 `Name,`

93 `Salary,`





94 `RANK() OVER (ORDER BY Salary DESC) AS SalaryRank`

95 `FROM`

96 `employee;`

97

98 • `SELECT`

Result Grid |   Filter Rows: | Export:  | Wrap Cell Content: 

	EmployeeID	Name	Salary	SalaryRank
▶	6	Asmita	20000000.00	1
	1	Emily Davis	50000.00	2
	3	Sophia Lee	45000.00	3
	5	Olivia Rodriguez	40000.00	4
	4	Daniel Clark	35000.00	5

Dense_rank()

Query 1 SQL File 3* SQL File 4* SQL File 5* x

Limit to 1000 rows

```
98 • SELECT
99     EmployeeID,
100     Name,
101     Salary,
102     DENSE_RANK() OVER (ORDER BY Salary DESC) AS DenseSalaryRank
103 FROM
104     employee;
105
106 • SELECT
107     EmployeeID,
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: [fA](#)

	EmployeeID	Name	Salary	DenseSalaryRank
▶	6	Asmita	20000000.00	1
	1	Emily Davis	50000.00	2
	3	Sophia Lee	45000.00	3
	5	Olivia Rodriguez	40000.00	4
	4	Daniel Clark	35000.00	5

Result 41 x

NTILE()

Limit to 1000 rows

```
104     employee;
105
106 • SELECT
107     EmployeeID,
108     Name,
109     Salary,
110     NTILE(4) OVER (ORDER BY Salary DESC) AS SalaryQuartile
111 FROM
112     employee;
113
```




Result Grid | Filter Rows: | Export: | Wrap Cell Content: [fA](#)

	EmployeeID	Name	Salary	SalaryQuartile
▶	6	Asmita	20000000.00	1
	1	Emily Davis	50000.00	1
	3	Sophia Lee	45000.00	2
	5	Olivia Rodriguez	40000.00	2
	4	Daniel Clark	35000.00	3

Result 42 x

ROW_NUMBER()

```
13
14
15 • SELECT
16     EmployeeID,
17     Name,
18     Salary,
19     ROW_NUMBER() OVER (ORDER BY Salary DESC) AS RowNumber
20 FROM
21     employee;
22
```

result Grid  Filter Rows: <input type="text"/> Export:  Wrap Cell Content: 				
	EmployeeID	Name	Salary	RowNumber
	6	Asmita	20000000.00	1
	1	Emily Davis	50000.00	2
	3	Sophia Lee	45000.00	3
	5	Olivia Rodriguez	40000.00	4
	4	Daniel Clark	35000.00	5

Result 43 ×