

Name: Krushnakumar Patle

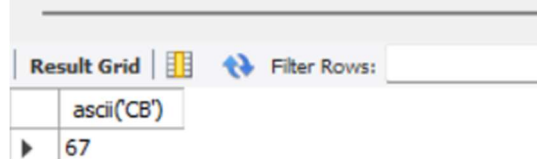
Email: [krishnapatle128@gmail.com](mailto:krishnapatle128@gmail.com)

Batch: Data Engineering Batch-1

## 1. String functions

Calculate ascii value in sql

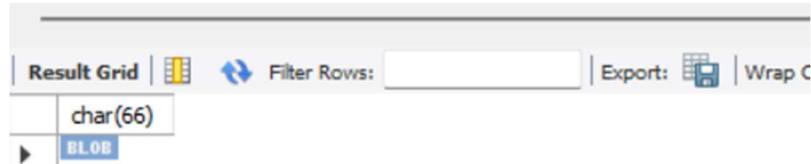
```
108 • select ascii('CB');
```



ascii('CB')
67

Calculate ascii of char

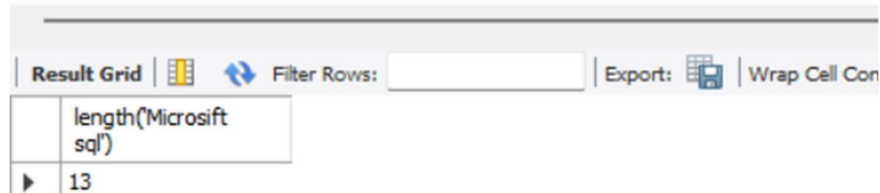
```
110 • select char(66);
111      -- /return ascii value to character/
```



char(66)
BLOB

Calculate length in sql

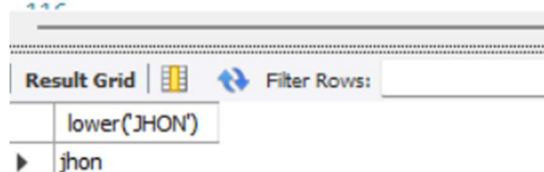
```
113 • select length('Microsift sql');
114
```



length('Microsift sql')
13

Use lower function in sql

```
115 • select lower('JHON');
```



lower('JHON')
jhon

Use replace function in sql

## Day 5 Assessment

```
117 • select replace('Microsoft sql','sql','server');
```

```
118
```

```
119 • select reverse('python');
```

Result Grid	Filter Rows:	Export:	Wrap Cell Cont
replace('Microsoft sql','sql','server')			
Microsoft server			

Use reverse function in sql

```
119 • select reverse('python');
```

```
---
```

Result Grid	Filter Rows:	Exp
reverse('python')		
nohtyp		

Use upper function in sql

```
121 • select upper('yourname');
```

```
122
```

Result Grid	Filter Rows:	Export:	V
upper('yourname')			
YOURNAME			

Use format function in sql

```
123 • SELECT FORMAT(136.564, 4);
```

```
124
```

Result Grid	Filter Rows:	Export:
FORMAT(136.564, 4)		
136.5640		

## 2. Date Functions

Calculate current datetime in sql

```
127 • SELECT CURRENT_TIMESTAMP() AS current_datetime;
```

```
128
```

Result Grid	Filter Rows:	Export:	Wrap Cell Content:
current_datetime			
2024-01-23 23:59:24			

Use DATE\_ADD function in sql

```
131 • SELECT DATE_ADD('2023-12-07', INTERVAL 2 MONTH) AS new_date;
132
```

new_date
2024-02-07

Calculate month from date

```
134 • SELECT MONTH('2008-05-22') AS month_value;
135 -- /return months value/
136
```

month_value
5

Calculate day

```
137 • select day ( '2023-05-30'); -- /return value of date of that particular day/
138
```

day ( '2023-05-30')
30

Calculate year

```
141 • select year ( '2023-05-3'); -- /return year value/
```

year ( '2023-05-3')
2023

### 3. Mathematical Functions

## Day 5 Assessment

```
145 • select abs(-101);  
146 -- /returns absolute value/  
147
```

Result Grid		Filter Rows:	Export:	Wrap Cell C
	abs(-101)			
▶	101			

```
148 • select sin(1.5);  
149 -- /returns angle in radians/  
150
```

Result Grid		Filter Rows:	Export:	Wrap
	sin(1.5)			
▶	0.9974949866040544			

```
151 • select ceiling(14.01);  
152 -- /returns the smallest or greater to the specified value/
```

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
	ceiling(14.01)			
▶	15			

```
154 • select exp(4.5);  
155 -- /returns the exponential value/  
156
```

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
	exp(4.5)			
▶	90.01713130052181			

```
157 • select floor(14.75);
```

Result Grid		Filter Rows:	Export:	Wrap C
	floor(14.75)			
▶	14			

## Day 5 Assessment

```
159 • select log(5.4);
160 -- /return logarithmic value/
161
162
```

Result Grid		Filter Rows:	Export:	Wrap Cell
	log(5.4)			
▶	1.6863989535702288			

### 4. Data cleaning and transformation

```
3 -- Create the table
4 • CREATE TABLE studentdata1 (
5     id INT PRIMARY KEY AUTO_INCREMENT,
6     name VARCHAR(255),
7     age INT,
8     grade VARCHAR(5)
9 );
10
11 -- Insert data into the table
12 • INSERT INTO studentdata1 (id, name, age, grade) VALUES (null, 'stella', 20, 'A+');
13 • INSERT INTO studentdata1 (id, name, age, grade) VALUES (1, 'appu', 20, 'A+');
14 • INSERT INTO studentdata1 (id, name, age, grade) VALUES (5, 'bob', 21, 'C');
15 • INSERT INTO studentdata1 (id, name, age, grade) VALUES (6, 'sunny', 21, null);
16 • INSERT INTO studentdata1 (id, name, age, grade) VALUES (7, null, 21, 'C');
17
21 /* STEP-1 ----> Deleting the duplicate data*/
22 • select name, count(name) as Actual_count from studentdata1
23     group by name
24     having count(name)>1;
```

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
	name	Actual_count		

## Day 5 Assessment

```
26 • with cte as
27 (
28     select name,
29     ROW_NUMBER() over (partition by name order by name desc) as row_no
30     from studentdata1)
31
32 select *from cte;
--
```

Result Grid	Filter Rows:	Export:	Wrap Cell Content:
	name	row_no	
	NULL	1	

```
31 -- Removing null values
32 • SELECT * FROM studentdata1;
33
```

Result Grid

Filter Rows:

Edit:

	id	name	age	grade
7	7	NULL	21	C
*	NULL	NULL	NULL	NULL

```
34 -- Selecting data where student name is null
35 • SELECT * FROM studentdata1
36 WHERE name IS NULL;
```

Result Grid					Filter Rows:		Edit:	
	id	name	age	grade				
▶	7	NULL	21	C				
*	NULL	NULL	NULL	NULL				

```
41 -- Updating null values where id is null
42 • SELECT * FROM studentdata1 WHERE id IS NULL;
```

Result Grid

Filter Rows:

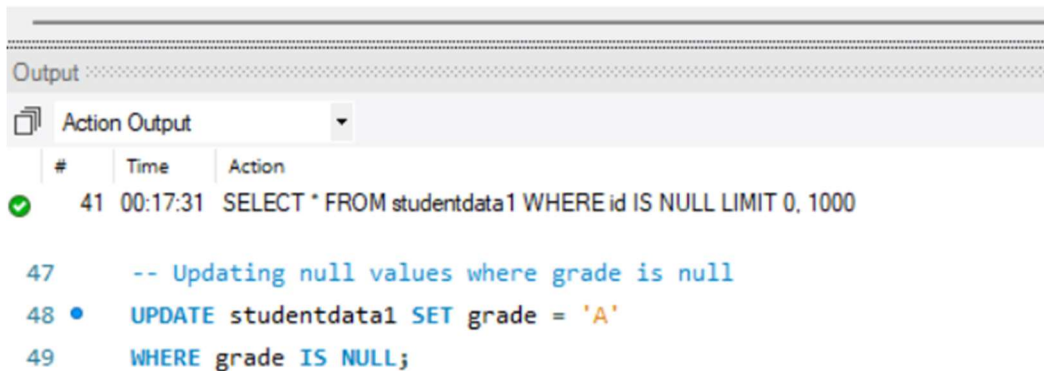
Edit

Export/Import

	id	name	age	grade
*	NULL	NULL	NULL	NULL

## Day 5 Assessment

```
44 • UPDATE studentdata1 SET id = 7
45 WHERE id IS NULL;
46
```



The screenshot shows a database interface with an 'Output' window. The window has a tab labeled 'Action Output'. Below the tab is a table with three columns: '#', 'Time', and 'Action'. The table contains one row with a green checkmark icon in the first column, the number '41' in the second column, the time '00:17:31' in the third column, and the SQL query 'SELECT \* FROM studentdata1 WHERE id IS NULL LIMIT 0, 1000' in the fourth column. Below the table, there are three lines of SQL code: a comment '-- Updating null values where grade is null', an update statement 'UPDATE studentdata1 SET grade = 'A'', and a where clause 'WHERE grade IS NULL;'.

#	Time	Action
✓ 41	00:17:31	SELECT * FROM studentdata1 WHERE id IS NULL LIMIT 0, 1000

```
47 -- Updating null values where grade is null
48 • UPDATE studentdata1 SET grade = 'A'
49 WHERE grade IS NULL;
```

## 5. Ranking in SQL

```
1 • CREATE TABLE ExamResult
2 (
3     StudentName VARCHAR(70),
4     Subject      VARCHAR(20),
5     Marks        INT
6 );
7
8 • INSERT INTO ExamResult VALUES('Lily','Maths',65);
9 • INSERT INTO ExamResult VALUES('Lily','Science',80);
10 • INSERT INTO ExamResult VALUES('Lily','english',70);
11 • INSERT INTO ExamResult VALUES('Isabella','Maths',50);
12 • INSERT INTO ExamResult VALUES('Isabella','Science',70);
13 • INSERT INTO ExamResult VALUES('Isabella','english',90);
14 • INSERT INTO ExamResult VALUES('Olivia','Maths',55);
15 • INSERT INTO ExamResult VALUES('Olivia','Science',60);
16 • INSERT INTO ExamResult VALUES('Olivia','english',89);
```

### Query 1 - ROW\_NUMBER:

- Assigns a unique row number to each row based on the ascending order of the **Marks** column.
- Rows with lower marks will have lower **RowNumber**, and vice versa.



## Day 5 Assessment

```
19 • SELECT Studentname, Subject, Marks, ROW_NUMBER() OVER(ORDER BY Marks) RowNumber
20 FROM ExamResult;
```

Result Grid				
		Filter Rows:	Export:	Wrap Cell Content:
	Studentname	Subject	Marks	RowNumber
▶	Isabella	Maths	50	1
	Olivia	Maths	55	2
	Olivia	Science	60	3
	Lily	Maths	65	4
	Lily	english	70	5
	Isabella	Science	70	6
	Lily	Science	80	7
	Olivia	english	89	8
	Isabella	english	90	9

### Query 2 - RANK:

- Assigns a rank to each row within each **Subject** based on the descending order of the **Marks** column.
- The **PARTITION BY** clause ensures that ranking is done separately for each **Subject**.

```
23 • SELECT
24     StudentName,
25     Subject,
26     Marks,
27     RANK() OVER (PARTITION BY Subject ORDER BY Marks DESC) AS SubjectRank
28 FROM
29     ExamResult;
```

Result Grid				
		Filter Rows:	Export:	Wrap Cell Content:
	StudentName	Subject	Marks	SubjectRank
▶	Isabella	english	90	1
	Olivia	english	89	2
	Lily	english	70	3
	Lily	Maths	65	1
	Olivia	Maths	55	2
	Isabella	Maths	50	3
	Lily	Science	80	1
	Isabella	Science	70	2
	Olivia	Science	60	3




### Query 3 - DENSE\_RANK:

- Similar to **RANK**, assigns a rank within each **Subject** based on the ascending order of the **Marks** column.



- However, unlike **RANK**, it does not leave gaps in the ranking when there are tied values.

```
32 • SELECT
33     StudentName,
34     Subject,
35     Marks,
36     DENSE_RANK() OVER (PARTITION BY Subject ORDER BY Marks) AS SubjectDenseRank
37 FROM
38     ExamResult;
```




Result Grid    Filter Rows: <input type="text"/>   Export:    Wrap Cell Content: 				
	StudentName	Subject	Marks	SubjectDenseRank
▶	Lily	english	70	1
	Olivia	english	89	2
	Isabella	english	90	3
	Isabella	Maths	50	1
	Olivia	Maths	55	2
	Lily	Maths	65	3
	Olivia	Science	60	1
	Isabella	Science	70	2
	Lily	Science	80	3

#### Query 4 - NTILE:

- **Meaning:** Divides the result set into equal-sized buckets (tiles) based on the descending order of the **Marks** column.
- In this case, it divides the data into two tiles (**NTILE(2)**), assigning each row to a tile based on its **Marks**.

## Day 5 Assessment

```
40 • SELECT *,
41      NTILE(2) OVER(
42      ORDER BY Marks DESC) as ntiles
43 FROM ExamResult
44 ORDER BY ntiles;
```

Result Grid    Filter Rows: <input type="text"/>   Export:    Wrap Cell Content: 				
	StudentName	Subject	Marks	ntiles
▶	Isabella	english	90	1
	Olivia	english	89	1
	Lily	Science	80	1
	Lily	english	70	1
	Isabella	Science	70	1
	Lily	Maths	65	2
	Olivia	Science	60	2
	Olivia	Maths	55	2
	Isabella	Maths	50	2

## 6. Stored procedure

### Table Creation and Data Insertion:

```
1 • CREATE TABLE Product
2   (ProductID INT, ProductName VARCHAR(100) );
3
4 • CREATE TABLE ProductDescription
5   (ProductID INT, ProductDescription VARCHAR(800) );
6
7 • INSERT INTO Product VALUES (680,'HL Road Frame - Black, 58')
8   ,(706,'HL Road Frame - Red, 58')
9   ,(707,'Sport-100 Helmet, Red');
10
11 • INSERT INTO ProductDescription VALUES (680,'Replacement mountain wheel for entry-level rider.')
12   ,(706,'Sturdy alloy features a quick-release hub.')
13   ,(707,'Aerodynamic rims for smooth riding.');
```

### Stored Procedure Creation:

## Day 5 Assessment

```
17  -- Create ProductInfoProcedure
18  DELIMITER //
19  CREATE PROCEDURE GetProductInfo(IN p_ProductID INT)
20  BEGIN
21      SELECT
22          p.ProductID,
23          p.ProductName,
24          pd.ProductDescription
25      FROM
26          Product p
27      JOIN
28          ProductDescription pd ON p.ProductID = pd.ProductID
29      WHERE
30          p.ProductID = p_ProductID;
31  END //
32
33  DELIMITER ;
```

Output				
Action Output				
#	Time	Action	Message	
60	00:33:14	CREATE TABLE ProductDescription (ProductID INT, ProductDescription VARCHAR(800))	0 row(s) affected	
61	00:33:19	CREATE TABLE ProductDescription (ProductID INT, ProductDescription VARCHAR(800))	Error Code: 1050. Table 'productdescription' already exists	
62	00:33:30	INSERT INTO Product VALUES (680,'HL Road Frame - Black, 58'),(706,'HL Road Frame - Red, 58'),(707,'Sp...	3 row(s) affected Records: 3 Duplicates: 0 Warnings: 0	
63	00:33:35	INSERT INTO ProductDescription VALUES (680,'Replacement mountain wheel for entry-level rider.),(706,'Stu...	3 row(s) affected Records: 3 Duplicates: 0 Warnings: 0	
64	00:34:59	CREATE PROCEDURE GetProductInfo(IN p_ProductID INT) BEGIN SELECT p.ProductID, p.Pro...	0 row(s) affected	

## Execute the Stored Procedure:

```
34  -- Execute the stored procedure with ProductID = 680
35  CALL GetProductInfo(680);
```

Result Grid			
Filter Rows:		Export:	Wrap Cell Content:
ProductID	ProductName	ProductDescription	
680	HL Road Frame - Black, 58	Replacement mountain wheel for entry-level rider.	