ASSIGNMENT TITLE

SQL FUNCTIONS ASSIGNMENT ANSWER EXPLANATION

Submitted by:

Name: Sachin Brijesh Yadav

Course: Data Analytics With AI

Batch: September 2025 Live (Pro Batch)

Registered Email ID: sachinyadav9496@gmail.com

Create Table

Query | Query History

```
1  CREATE TABLE Student_Performance (
2      student_id INT PRIMARY KEY,
3      name VARCHAR(50),
4      course VARCHAR(30),
5      score INT,
6      attendance INT,
7      mentor VARCHAR(50),
8      join_date DATE,
9      city VARCHAR(50)
10 );
11
```

Data Output | Messages | Notifications

```
CREATE TABLE

Query returned successfully in 4 secs 113 msec.
```

Insert Data

Query | Query History

```
1  INSERT INTO Student_Performance
2  (student_id, name, course, score, attendance, mentor, join_date, city)
3  VALUES
4  (101, 'Aarav Mehta', 'Data Science', 88, 92, 'Dr. Sharma', '2023-06-12', 'Mumbai'),
5  (102, 'Riya Singh', 'Data Science', 76, 85, 'Dr. Sharma', '2023-07-01', 'Delhi'),
6  (103, 'Kabir Khanna', 'Python', 91, 96, 'Ms. Nair', '2023-06-20', 'Mumbai'),
7  (104, 'Tanvi Patel', 'SQL', 84, 89, 'Mr. Iyer', '2023-05-30', 'Bengaluru'),
8  (105, 'Ayesha Khan', 'Python', 67, 81, 'Ms. Nair', '2023-07-10', 'Hyderabad'),
9  (106, 'Dev Sharma', 'SQL', 73, 78, 'Mr. Iyer', '2023-05-28', 'Pune'),
10 (107, 'Arjun Verma', 'Tableau', 95, 98, 'Ms. Kapoor', '2023-06-15', 'Delhi'),
11 (108, 'Meera Pillai', 'Tableau', 82, 87, 'Ms. Kapoor', '2023-06-18', 'Kochi'),
12 (109, 'Nikhil Rao', 'Data Science', 79, 82, 'Dr. Sharma', '2023-07-05', 'Chennai'),
```

Data Output | Messages | Notifications

```
INSERT 0 15

Query returned successfully in 15 secs 143 msec.
```

QUESTION 1 – Create a ranking of students based on score (highest first).

Concept:

Use **RANK()** or **DENSE_RANK()** window function ordered by score descending.

SQL Query:

Query | Query History

```
SELECT
    student_id,
    name,
    score,
    RANK() OVER (ORDER BY score DESC) AS score_rank
FROM Student_Performance;
```

Output:

Data Output | Messages | Notifications

	student_id [PK] integer	name character varying (50)	score integer	score_rank bigint
1	107	Arjun Verma	95	1
2	114	Nikita Joshi	93	2
3	110	Priya Desai	92	3
4	103	Kabir Khanna	91	4
5	113	Rohan Gupta	89	5
6	101	Aarav Mehta	88	6
7	111	Siddharth Jain	85	7
8	104	Tanvi Patel	84	8
9	108	Meera Pillai	82	9
10	109	Nikhil Rao	79	10
11	102	Riya Singh	76	11
12	112	Sneha Kulkarni	74	12
13	106	Dev Sharma	73	13
14	115	Yuvraj Rao	71	14
15	105	Ayesha Khan	67	15

Total rows: 15 | Query complete 00:00:06.805

Explanation:

- Students with the same score get the same rank.
- Ranking starts from highest score → lowest.

QUESTION 2 – Show each student's score and previous student's score (based on score order).

Concept:

Use **LAG()** window function.

SQL Query:

Query | Query History

```sql
SELECT
    student_id,
    name,
    score,
    LAG(score, 1) OVER (ORDER BY score DESC) AS previous_score
FROM Student_Performance;
```

Output:

Data Output | Messages | Notifications

	student_id [PK] integer	name character varying (50)	score integer	previous_score integer
1	107	Arjun Verma	95	[null]
2	114	Nikita Joshi	93	95
3	110	Priya Desai	92	93
4	103	Kabir Khanna	91	92
5	113	Rohan Gupta	89	91
6	101	Aarav Mehta	88	89
7	111	Siddharth Jain	85	88
8	104	Tanvi Patel	84	85
9	108	Meera Pillai	82	84
10	109	Nikhil Rao	79	82
11	102	Riya Singh	76	79
12	112	Sneha Kulkarni	74	76
13	106	Dev Sharma	73	74
14	115	Yuvraj Rao	71	73
15	105	Ayesha Khan	67	71

Total rows: 15 | Query complete 00:00:15.493

Explanation:

- `LAG()` extracts the score of the student immediately above in ranking.
- The top student has NULL as previous score.

QUESTION 3 – Convert all student names to uppercase AND extract month name from join_date.

SQL Query:

Query | Query History

```sql
SELECT
    UPPER(name) AS upper_name,
    TO_CHAR(join_date, 'Month') AS join_month
FROM Student_Performance;
```

Output:

Data Output | Messages | Notifications

	upper_name text	join_month text
1	AARAV MEHTA	June
2	RIYA SINGH	July
3	KABIR KHANNA	June
4	TANVI PATEL	May
5	AYESHA KHAN	July
6	DEV SHARMA	May
7	ARJUN VERMA	June
8	MEERA PILLAI	June
9	NIKHIL RAO	July
10	PRIYA DESAI	May
11	SIDDHARTH JAIN	July
12	SNEHA KULKAR...	June
13	ROHAN GUPTA	May
14	NIKITA JOSHI	June
15	YUVRAJ RAO	July

Total rows: 15 | Query complete 00:00:23.794

Explanation:

- `UPPER()` → converts name to uppercase.
- `MONTHNAME()` → converts date to month (e.g., June).

QUESTION 4 – Show each student's name and the next student's attendance.

Concept:

Use **LEAD()** function ordered by attendance ascending.

SQL Query:

Query | Query History

```
SELECT
    name,
    attendance,
    LEAD(attendance, 1) OVER (ORDER BY attendance) AS next_attendance
FROM Student_Performance;
```

Output:

Data Output | Messages | Notifications

SQL

	name character varying (50)	attendance integer	next_attendance integer
1	Dev Sharma	78	81
2	Ayesha Khan	81	82
3	Nikhil Rao	82	83
4	Sneha Kulkarni	83	84
5	Yuvraj Rao	84	85
6	Riya Singh	85	87
7	Meera Pillai	87	89
8	Tanvi Patel	89	90
9	Siddharth Jain	90	91
10	Rohan Gupta	91	92
11	Aarav Mehta	92	94
12	Priya Desai	94	96
13	Kabir Khanna	96	97
14	Nikita Joshi	97	98
15	Arjun Verma	98	[null]

Total rows: 15 | Query complete 00:00:13.401

Explanation:

Shows how a student's attendance compares to the next student.

QUESTION 5 – Assign students into 4 performance groups using NTILE().

SQL Query:

Query | Query History

```sql
SELECT
    student_id,
    name,
    score,
    NTILE(4) OVER (ORDER BY score DESC) AS performance_group
FROM Student_Performance;
```

Output:

Data Output | Messages | Notifications

	student_id [PK] integer	name character varying (50)	score integer	performance_group integer
1	107	Arjun Verma	95	1
2	114	Nikita Joshi	93	1
3	110	Priya Desai	92	1
4	103	Kabir Khanna	91	1
5	113	Rohan Gupta	89	2
6	101	Aarav Mehta	88	2
7	111	Siddharth Jain	85	2
8	104	Tanvi Patel	84	2
9	108	Meera Pillai	82	3
10	109	Nikhil Rao	79	3
11	102	Riya Singh	76	3
12	112	Sneha Kulkarni	74	3
13	106	Dev Sharma	73	4
14	115	Yuvraj Rao	71	4
15	105	Ayesha Khan	67	4

Total rows: 15 | Query complete 00:00:15.496

Explanation:

Divides class into 4 equal groups:

- Group 1 = top performers
- Group 4 = lowest performers

QUESTION 6 – For each course, assign row number based on attendance (highest first).

SQL Query:

Query | Query History

```
1  SELECT
2      student_id,
3      name,
4      course,
5      attendance,
6      ROW_NUMBER() OVER (
7          PARTITION BY course
8          ORDER BY attendance DESC
9      ) AS attendance_rank
10 FROM Student_Performance;
11
```

Output:

Data Output | Messages | Notifications

Showing rows: 1 to 15

	student_id [PK] integer	name character varying (50)	course character varying (30)	attendance integer	attendance_rank bigint
1	114	Nikita Joshi	Data Science	97	1
2	101	Aarav Mehta	Data Science	92	2
3	102	Riya Singh	Data Science	85	3
4	109	Nikhil Rao	Data Science	82	4
5	103	Kabir Khanna	Python	96	1
6	111	Siddharth Jain	Python	90	2
7	115	Yuvraj Rao	Python	84	3
8	105	Ayesha Khan	Python	81	4
9	110	Priya Desai	SQL	94	1
10	113	Rohan Gupta	SQL	91	2
11	104	Tanvi Patel	SQL	89	3
12	106	Dev Sharma	SQL	78	4
13	107	Arjun Verma	Tableau	98	1
14	108	Meera Pillai	Tableau	87	2
15	112	Sneha Kulkarni	Tableau	83	3

Total rows: 15 | Query complete 00:00:09.007

Explanation:

`PARTITION BY course` → ranking resets for each course.

QUESTION 7 – Number of days each student has been enrolled (Assume today = '2025-01-01').

SQL Query:

Query | Query History

```
1
2  SELECT
3      student_id,
4      name,
5      (DATE '2025-01-01' - join_date) AS days_enrolled
6  FROM Student_Performance;
7
```

Output:

Data Output | Messages | Notifications

	student_id [PK] integer	name character varying (50)	days_enrolled integer
1	101	Aarav Mehta	569
2	102	Riya Singh	550
3	103	Kabir Khanna	561
4	104	Tanvi Patel	582
5	105	Ayesha Khan	541
6	106	Dev Sharma	584
7	107	Arjun Verma	566
8	108	Meera Pillai	563
9	109	Nikhil Rao	546
10	110	Priya Desai	585
11	111	Siddharth Jain	549
12	112	Sneha Kulkarni	571
13	113	Rohan Gupta	587
14	114	Nikita Joshi	576
15	115	Yuvraj Rao	539

Total rows: 15 | Query complete 00:00:12.892

Explanation:

`DATEDIFF()` subtracts join_date from the given date.

QUESTION 8 – Format join_date as “Month Year” (e.g., “June 2023”).

SQL Query:

Query | Query History

```sql
SELECT
    name,
    TO_CHAR(join_date, 'FMMonth YYYY') AS formatted_date
FROM Student_Performance;
```

Output:

Data Output | Messages | Notifications

	name character varying (50)	formatted_date text
1	Aarav Mehta	June 2023
2	Riya Singh	July 2023
3	Kabir Khanna	June 2023
4	Tanvi Patel	May 2023
5	Ayesha Khan	July 2023
6	Dev Sharma	May 2023
7	Arjun Verma	June 2023
8	Meera Pillai	June 2023
9	Nikhil Rao	July 2023
10	Priya Desai	May 2023
11	Siddharth Jain	July 2023
12	Sneha Kulkarni	June 2023
13	Rohan Gupta	May 2023
14	Nikita Joshi	June 2023
15	Yuvraj Rao	July 2023

Total rows: 15 | Query complete 00:00:08.018

QUESTION 9 – Replace city ‘Mumbai’ with ‘MUM’.

SQL Query:

Query | Query History

```sql
SELECT
    name,
    city,
    REPLACE(city, 'Mumbai', 'MUM') AS updated_city
FROM Student_Performance;
```

Output:

Data Output | Messages | Notifications

	name character varying (50)	city character varying (50)	updated_city text
1	Aarav Mehta	Mumbai	MUM
2	Riya Singh	Delhi	Delhi
3	Kabir Khanna	Mumbai	MUM
4	Tanvi Patel	Bengaluru	Bengaluru
5	Ayesha Khan	Hyderabad	Hyderabad
6	Dev Sharma	Pune	Pune
7	Arjun Verma	Delhi	Delhi
8	Meera Pillai	Kochi	Kochi
9	Nikhil Rao	Chennai	Chennai
10	Priya Desai	Bengaluru	Bengaluru
11	Siddharth Jain	Mumbai	MUM
12	Sneha Kulkarni	Pune	Pune
13	Rohan Gupta	Delhi	Delhi
14	Nikita Joshi	Bengaluru	Bengaluru
15	Yuvraj Rao	Hyderabad	Hyderabad

Total rows: 15 | Query complete 00:00:16.596

QUESTION 10 – For each course, find the highest score using FIRST_VALUE().

SQL Query:

Query | Query History

```sql
SELECT
    student_id,
    name,
    course,
    score,
    FIRST_VALUE(score) OVER (
        PARTITION BY course
        ORDER BY score DESC
    ) AS highest_course_score
FROM Student_Performance;
```

Output:

Data Output | Messages | Notifications

Showing rows: 1 to 15

	student_id [PK] integer	name character varying (50)	course character varying (30)	score integer	highest_course_score integer
1	114	Nikita Joshi	Data Science	93	93
2	101	Aarav Mehta	Data Science	88	93
3	109	Nikhil Rao	Data Science	79	93
4	102	Riya Singh	Data Science	76	93
5	103	Kabir Khanna	Python	91	91
6	111	Siddharth Jain	Python	85	91
7	115	Yuvraj Rao	Python	71	91
8	105	Ayesha Khan	Python	67	91
9	110	Priya Desai	SQL	92	92
10	113	Rohan Gupta	SQL	89	92
11	104	Tanvi Patel	SQL	84	92
12	106	Dev Sharma	SQL	73	92
13	107	Arjun Verma	Tableau	95	95
14	108	Meera Pillai	Tableau	82	95
15	112	Sneha Kulkarni	Tableau	74	95

Total rows: 15 | Query complete 00:00:15.044