**OUBT Week-1 – Day 5: SQL & Data Modeling (AWS RDS)**

**NAME : LIKHITH SASANK UPPALAPATI VENKATA**

**Overview:**

Day 5 focused on setting up and working with a live PostgreSQL database on AWS RDS. I created the instance, configured IAM access, and connected through DBeaver to build tables and run SQL queries on sample taxi data. I also checked RDS metrics in CloudWatch to confirm query activity and understand how CPU and connections change when the database is in use. This helped me connect SQL practice with real AWS database monitoring and management

**Objective:**  
Design a simple relational schema on AWS RDS PostgreSQL and practice SQL operations on sample NYC Taxi Trips data.

**Steps Performed:**

1. Created an **AWS RDS PostgreSQL** instance under Free Tier with public access enabled.

A screenshot of a computer

AI-generated content may be incorrect.

1. Configured inbound rules (port 5432) to allow access from local machine.

A screenshot of a computer

AI-generated content may be incorrect.

1. Connected to the RDS instance through **DBeaver** using the database endpoint.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

1. Created a new database nyc\_taxi\_db.

A screenshot of a computer

AI-generated content may be incorrect.

1. Designed schema with three tables — vendors, payment\_types, and nyc\_taxi\_trips — following a **Star Schema** model.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

1. Inserted sample trip data and reference data for vendors and payment types.
2. Executed multiple **SQL queries** covering JOIN, GROUP BY, aggregations, filtering, window functions, and CTEs to analyze trip records.

**SELECT** *v*.vendor\_name,

**COUNT**(\*) **AS** *total\_trips*,

**SUM**(fare\_amount) **AS** *total\_fare*

**FROM** nyc\_taxi\_trips *t*

**JOIN** vendors *v* **ON** *t*.vendor\_id = *v*.vendor\_id

**GROUP** **BY** *v*.vendor\_name;

**SELECT** *p*.payment\_name,

**ROUND**(**AVG**(fare\_amount),2) **AS** *avg\_fare*

**FROM** nyc\_taxi\_trips *t*

**JOIN** payment\_types *p* **ON** *t*.payment\_type = *p*.payment\_type

**GROUP** **BY** *p*.payment\_name;

**SELECT** trip\_id, vendor\_id, fare\_amount

**FROM** nyc\_taxi\_trips

**ORDER** **BY** fare\_amount **DESC**

**LIMIT** 3;

**SELECT** trip\_id, vendor\_id, fare\_amount,

**RANK**() **OVER**(**ORDER** **BY** fare\_amount **DESC**) **AS** *fare\_rank*

**FROM** nyc\_taxi\_trips;

1. Verified query results and captured screenshots of successful execution in DBeave

**CloudWatch & RDS Monitoring Test**

To verify my AWS RDS instance activity, I ran multiple SQL queries from DBeaver and monitored performance in the RDS dashboard. The CloudWatch graphs showed clear spikes in **CPU utilization**, **DB load**, and **active connections** while queries were executing. This confirmed that my PostgreSQL instance was processing live workloads.

This exercise helped me understand how CloudWatch metrics can be used to monitor database health and query performance in real time.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

**Key Learnings:**

* Gained hands-on experience in **AWS RDS setup**, database connectivity, schema creation, and SQL querying through DBeaver.
* Configured **IAM permissions and security groups** to enable secure database access.
* Understood how **Star Schema modeling** supports analytical queries and relational design.
* Practiced organizing and structuring data for better readability, consistency, and performance.