1. Database Design

The database is designed to store and query skier lift ride data efficiently. The schema is optimized to support the following queries:

Skier Activity: Retrieve days skied, vertical totals, and lifts used for a specific skier.

Resort Activity: Count unique skiers visiting a resort on a specific day.

Schema

Table Name: LiftRides

| Column Name | Data Type | Description |
| --- | --- | --- |
| id | INT | Auto-increment primary key. |
| skierID | INT | Unique ID of the skier. |
| resortID | INT | Unique ID of the resort. |
| seasonID | VARCHAR(10) | Season identifier (e.g., "2024"). |
| dayID | INT | Ski day identifier (e.g., 1 for Day 1). |
| time | INT | Time of the lift ride in minutes. |
| liftID | INT | Unique ID of the lift used. |
| vertical | INT | Precomputed vertical value (liftID \* 10). |

Indexes

1. idx\_skier\_season\_day:

Fields: (skierID, seasonID, dayID)

Optimizes queries by skier, season, and day.

1. idx\_resort\_day:

Fields: (resortID, dayID)

Optimizes queries for skier activity at a specific resort on a given day.

1. idx\_skierID:

Fields: (skierID)

Speeds up queries that involve only skierID.

2. Deployment Topology on AWS

Database Deployment

* AWS RDS (Relational Database Service):

Engine: MySQL 8.0

Instance Type: db.t3.micro

Storage: General Purpose SSD (gp2), starting with 20 GB.

* High Availability:

Multi-AZ deployment ensures automatic failover to a standby instance in case of a failure.

Automated backups are configured with a retention period of 7 days.

* Read Scalability:

Read replicas can be added if read-heavy workloads are observed.

Consumer Deployment

* AWS EC2:

The LiftRideConsumer application runs on an EC2 instance.

Instance Type: t3.medium (2 vCPUs, 4 GB RAM) for moderate message throughput.

The EC2 instance and RDS database are placed in the same VPC and subnet for secure and low-latency communication.

Security and Access

* Security Groups:

Restrict database access to only the EC2 instance running the consumer application.

Open ports for RabbitMQ communication and administrative tasks.

3. Scaling and Future Enhancements

* Database Scaling:

Vertical scaling (upgrading the RDS instance type) for higher throughput. At first I used t2 micro, which limited to 50 connections, then scaled up to t3 small.