Here's a detailed explanation for all your questions with examples and step-by-step explanations. Let's cover each question in detail, starting with table creation, optimization techniques, and other related concepts.

---

1. Database Architecture for Express Cargo Module

Entities: Flight, Cargo, Pricing, and Revenue

To design a database for an express cargo module, the entities and their relationships must be well-defined.

Tables and Relationships:

1. **Flight Table**:
   - Stores flight details.
   - `**FlightID**` is the primary key.
2. **Cargo Table**:
   - Stores cargo details, linked to flights.
   - `**CargoID**` is the primary key.
   - `**FlightID**` is a foreign key referencing `**FlightID**` in the `**Flight**` table.
3. **Pricing Table**:
   - Stores pricing details for cargo types.
   - `**PricingID**` is the primary key.
4. **Revenue Table**:
   - Tracks revenue for flights based on cargo bookings.
   - `**RevenueID**` is the primary key.

Schema for Tables

1. **Flight Table:**

```
sql


  CREATE TABLE Flight (
      FlightID INT AUTO_INCREMENT PRIMARY KEY,
      FlightNumber VARCHAR(20) NOT NULL,
      Destination VARCHAR(50),
      DepartureTime DATETIME
  );
```

2. **Cargo Table:**

```
sql


  CREATE TABLE Cargo (
      CargoID INT AUTO_INCREMENT PRIMARY KEY,
      CargoType VARCHAR(50),
      Weight DECIMAL(10, 2),
```

```
    Volume DECIMAL(10, 2),
    FlightID INT,
    FOREIGN KEY (FlightID) REFERENCES Flight(FlightID)
);
```

3. **Pricing Table:**

```sql
CREATE TABLE Pricing (
    PricingID INT AUTO_INCREMENT PRIMARY KEY,
    CargoType VARCHAR(50),
    PricePerKg DECIMAL(10, 2),
    PricePerVolume DECIMAL(10, 2)
);
```

4. **Revenue Table:**

```sql
CREATE TABLE Revenue (
    RevenueID INT AUTO_INCREMENT PRIMARY KEY,
    FlightID INT,
    TotalRevenue DECIMAL(10, 2),
    FOREIGN KEY (FlightID) REFERENCES Flight(FlightID)
);
```

Table Relationships:

- **Flight → Cargo (One-to-Many):** A flight can carry multiple cargos.
- **Pricing → Cargo (One-to-Many):** Each cargo type has a specific pricing.
- **Flight → Revenue (One-to-One):** Each flight has one revenue record.

---

2. Techniques to Optimize Database Operations

1. **Indexing**:
   - Speeds up query performance.
   - Example: Index the `**FlightNumber**` column for quick searches.

```sql
CREATE INDEX idx_flight_number ON Flight(FlightNumber);
```

2. **Partitioning**:
   - Divides a table into smaller, manageable pieces.
   - Example: Partition `**Flight**` data by `**Destination**`.
3. **Normalization**:
   - Reduces data redundancy by splitting tables.

4. **Denormalization**:
    - Combines frequently joined tables to reduce query time.
5. **Caching**:
    - Stores frequently accessed data in memory.
6. **Query Optimization**:
    - Use `EXPLAIN` to analyze queries.

```sql
EXPLAIN SELECT * FROM Cargo WHERE Weight > 100;
```

---

3. Database Partitioning

Partitioning splits a large table into smaller, manageable chunks based on a column, such as `DATE` or `LOCATION`. This helps with:

- **Query Performance**: Only relevant partitions are scanned.
- **Maintenance**: Easier to manage subsets of data.

Example:

Partition the `Flight` table by `Destination`:

```sql
CREATE TABLE Flight (
    FlightID INT AUTO_INCREMENT PRIMARY KEY,
    FlightNumber VARCHAR(20) NOT NULL,
    Destination VARCHAR(50),
    DepartureTime DATETIME
) PARTITION BY LIST COLUMNS (Destination) (
    PARTITION p_north VALUES IN ('NYC', 'BOS'),
    PARTITION p_west VALUES IN ('SFO', 'LAX')
);
```

---

4. Table Schema with Primary and Foreign Keys

1. **Primary Key**:
    - Uniquely identifies a record in a table.
    - Example: `FlightID` in the `Flight` table.
2. **Foreign Key**:
    - Links records in one table to another.
    - Example: `FlightID` in the `Cargo` table references `FlightID` in the `Flight` table.

Example Schema:

```sql
CREATE TABLE Flight (
    FlightID INT AUTO_INCREMENT PRIMARY KEY,
    FlightNumber VARCHAR(20) NOT NULL
);

CREATE TABLE Cargo (
    CargoID INT AUTO_INCREMENT PRIMARY KEY,
    FlightID INT,
    FOREIGN KEY (FlightID) REFERENCES Flight(FlightID)
);
```

---

5. Index a Query to Find Maximum Flight Number for a Destination

Query:

```sql
SELECT MAX(FlightNumber) FROM Flight WHERE Destination = 'NYC';
```

Index for Optimization:

```sql
CREATE INDEX idx_destination ON Flight(Destination);
```

---

6. Select MAX(FlightNumber) for 2 Million Records with Group By

Query:

```sql
SELECT Destination, MAX(FlightNumber)
FROM Flight
WHERE DepartureTime > '2024-01-01'
GROUP BY Destination;
```

Optimization Techniques:

1. **Index on `DepartureTime`:**

```sql
CREATE INDEX idx_departure_time ON Flight(DepartureTime);
```

2. **Partitioning by `Destination`.**

---

7. Indexing in MySQL

What Are Indexes? Indexes are data structures that improve query performance by allowing faster lookups.

Example:

Index the `FlightNumber` column:

```sql
CREATE INDEX idx_flight_number ON Flight(FlightNumber);
```

How Indexing Works:

Internally, MySQL uses **B-trees** or **hash tables** for indexes. It creates a sorted data structure, reducing the number of rows scanned during a query.

---

8. Indexing vs. Triggers

Feature	Indexing	Triggers
Purpose	Speeds up query performance.	Automates actions based on events.
Usage	Improves `SELECT` queries.	Used for enforcing rules or logging changes.
Example	Index on `FlightNumber`.	Trigger on `INSERT` to log changes.

---

9. Database Transactions

What Are Transactions? A **transaction** is a group of operations performed as a single unit of work, ensuring data consistency.

Properties (ACID):

1. **Atomicity**: All operations succeed or none.
2. **Consistency**: Maintains database integrity.
3. **Isolation**: Transactions don't interfere with each other.
4. **Durability**: Changes persist even after system failures.

Example:

```sql
START TRANSACTION;
UPDATE Seats SET Available = Available - 1 WHERE FlightID = 101;
```