# **Introduction**

Indexes are essential for optimizing query performance in data warehousing. They allow the database to quickly locate and retrieve data without scanning entire tables. This report focuses on the types of indexes used in our data warehouse model, the specific columns they are applied to, and the reasons for their usage.

**1. B-Tree Indexes**

Columns:

Primary Key Columns: dim\_date.date\_id, dim\_time.time\_id, dim\_airport.airport\_id, dim\_aircraft.aircraft\_id, dim\_passenger.passenger\_id.

Foreign Key Columns: fact\_flight\_activity.passenger\_id, fact\_customer\_feedback.aircraft\_id, agg\_flight\_trip.origin\_airport\_id.

Why: Primary keys are used for unique identification, and B-Tree indexes ensure fast lookups.

Foreign keys are frequently used in JOIN operations, and indexing them improves query performance.

**2. Bitmap Indexes**

Columns:

Low-Cardinality Columns: dim\_date.is\_holiday, dim\_date.is\_weekend, fact\_flight\_activity.flight\_completed.

Why: These columns have few distinct values, making bitmap indexes efficient for filtering and aggregation.

Bitmap indexes are ideal for data warehousing queries involving multiple low-cardinality columns.

**3. Composite Indexes**

Columns:

Multi-Column Combinations: dim\_date(year, month, day\_of\_month), fact\_flight\_activity(scheduled\_departure\_date\_id, scheduled\_departure\_time\_id).

Why: Queries often filter or sort on multiple columns simultaneously.

Composite indexes reduce the need for multiple single-column indexes and improve query performance.

**4. Full-Text Indexes**

Columns:

Large Text Fields: fact\_customer\_feedback.review\_body.

Why: Full-text indexes enable efficient keyword-based searches within large text fields.

Regular B-Tree indexes are not suitable for partial matches or keyword searches.

**5. Non-Clustered Indexes**

Columns:

Frequently Filtered Columns: fact\_customer\_feedback.overall\_rating, fact\_flight\_activity.base\_fare\_revenue.

Why: Non-clustered indexes are used for columns frequently used in WHERE clauses or JOIN conditions.

A table can have multiple non-clustered indexes, making them versatile for various query patterns.

**6. Bitmap Join Indexes**

Columns:

Join Columns: fact\_flight\_activity.origin\_airport\_id (JOIN with dim\_airport.airport\_id).

Why: Bitmap join indexes precompute the results of joins between tables, speeding up star schema queries.

They are particularly useful in data warehousing for optimizing join-heavy queries.