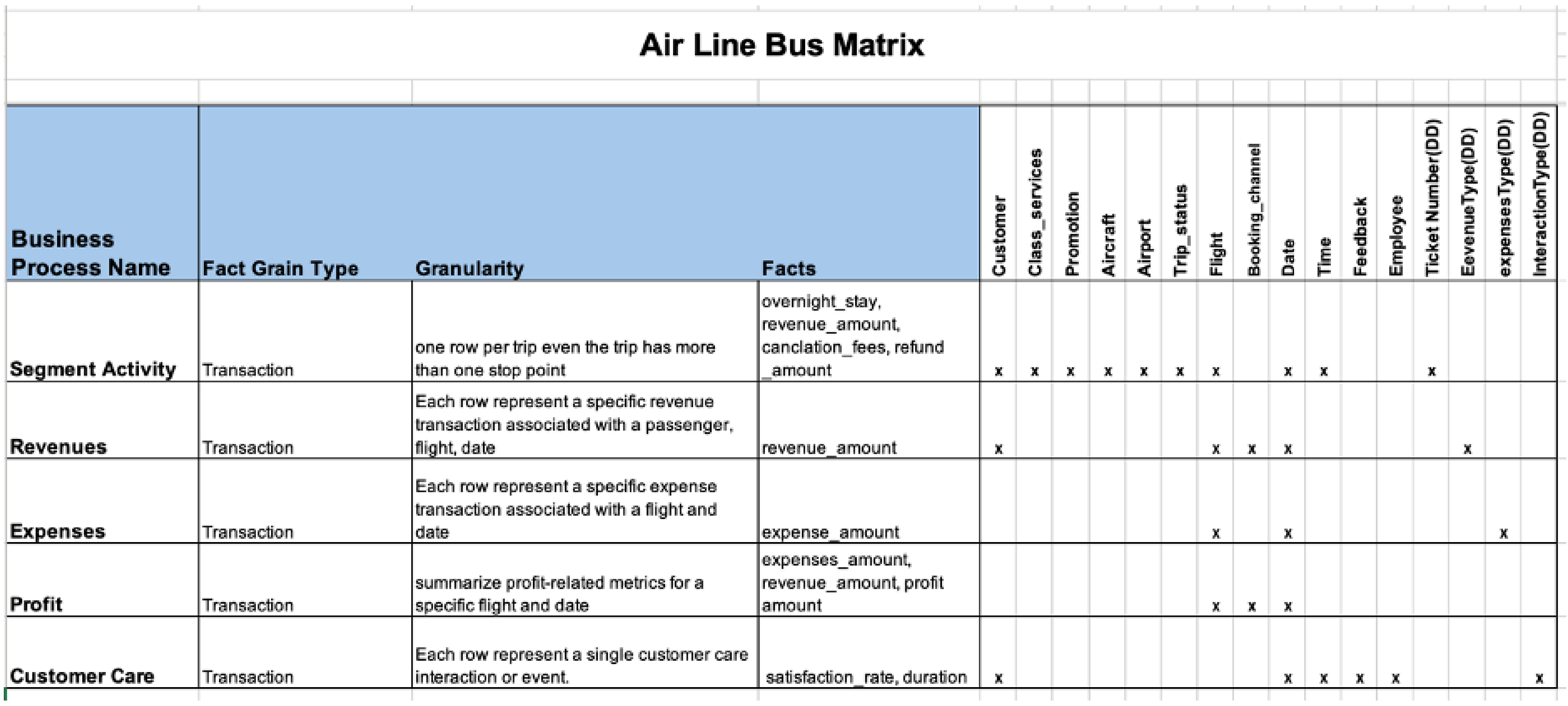
# AirLine Case Study

| | |
|---|---|
| Status | Complete |
| Course | Data Warehouse |
| Type | project |
| Due Date | @March 12, 2025 |
| Days Left | 1 days |
| Favorite | □ |
| Tags | |

## AirLine Project



Air Line Bus Matrix

| Business Process Name | Fact Grain Type | Granularity | Facts | Customer | Class_services | Promotion | Aircraft | Airport | Trip_status | Flight | Booking_channel | Date | Time | Feedback | Employee | Ticket Number(DD) | EevenueType(DD) | expensesType(DD) | InteractionType(DD) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Segment Activity | Transaction | one row per trip even the trip has more than one stop point | overnight_stay, revenue_amount, canclation_fees, refund _amount | x | x | x | x | x | x | x | | x | x | | | x | | | |
| Revenues | Transaction | Each row represent a specific revenue transaction associated with a passenger, flight, date | revenue_amount | x | | | | | | x | x | x | | | | | x | | |
| Expenses | Transaction | Each row represent a specific expense transaction associated with a flight and date | expense_amount | | | | | | | x | | x | | | | | | x | |
| Profit | Transaction | summarize profit-related metrics for a specific flight and date | expenses_amount, revenue_amount, profit amount | | | | | | | x | x | x | | | | | | | |
| Customer Care | Transaction | Each row represent a single customer care interaction or event. | satisfaction_rate, duration | x | | | | | | | | x | x | x | x | | | | x |

### Segment Activity

- **Business Process**: Tracks trip-related activities, including revenue, cancellations, refunds, and overnight stays.
- **Grain**: One row per trip, even if the trip has multiple stop points.
- **Fact Table**: SegmentActivityFact
  - **Measures**:
    - **overnight_stay**: Number of overnight stays during the trip.
    - **revenue_amount**: Revenue generated from the trip.
    - **cancellation_fees**: Fees charged for cancellations.
    - **refund_amount**: Amount refunded to the customer.
  - **Dimensions**:
    - **passenger_id**: Links to the customer_dim table to identify the passenger.
    - **class_services_id**: Links to the class_services_dim table to track the class of service purchased and flown.
    - **promotion_id**: Links to the promotion_dim table to track any promotions applied.
    - **flight_id**: Links to the flight_dim table to identify the flight.
    - **status_id**: Links to the trip_status_dim table to track the trip status (e.g., confirmed, canceled).
    - **date_id:** Links to the date_dim table to track the date of the trip.
    - **time_id**: Links to the time_dim table to track the time of the trip.

## Revenues

- **Business Process**: Tracks revenue transactions associated with passengers, flights, and dates.
- **Grain**: One row per revenue transaction.
- **Fact Table**: RevenueFact
    - **Measures**:
        - **revenue_amount**: Revenue generated from the transaction.
    - **Dimensions**:
        - **passenger_id**: Links to the customer_dim table to identify the passenger.
        - **date_id**: Links to the date_dim table to track the date of the transaction.
        - **flight_id**: Links to the flight_dim table to identify the flight.
        - **promotion_id**: Links to the promotion_dim table to track any promotions applied.
        - **booking_channel_id**: Links to the booking_channel_dim table to track the booking channel used.
        - **Revenue_type(DD)** : a degenerating dimension that refer to where the revenue came from (ticket sales, bagging fess, etc)

## Expenses

- **Business Process**: Tracks expense transactions associated with flights and dates.
- **Grain**: One row per expense transaction.
- **Fact Table**: ExpensesFact
    - **Measures**:
        - **expense_amount**: Expense incurred for the flight.
    - **Dimensions**:
        - **date_id**: Links to the date_dim table to track the date of the expense.
        - **flight_id**: Links to the flight_dim table to identify the flight.
        - **Expense_type**: a degenerating dimension that refer to the what type of expense this amount paid for (e.g., fuel, crew costs)

## Profit

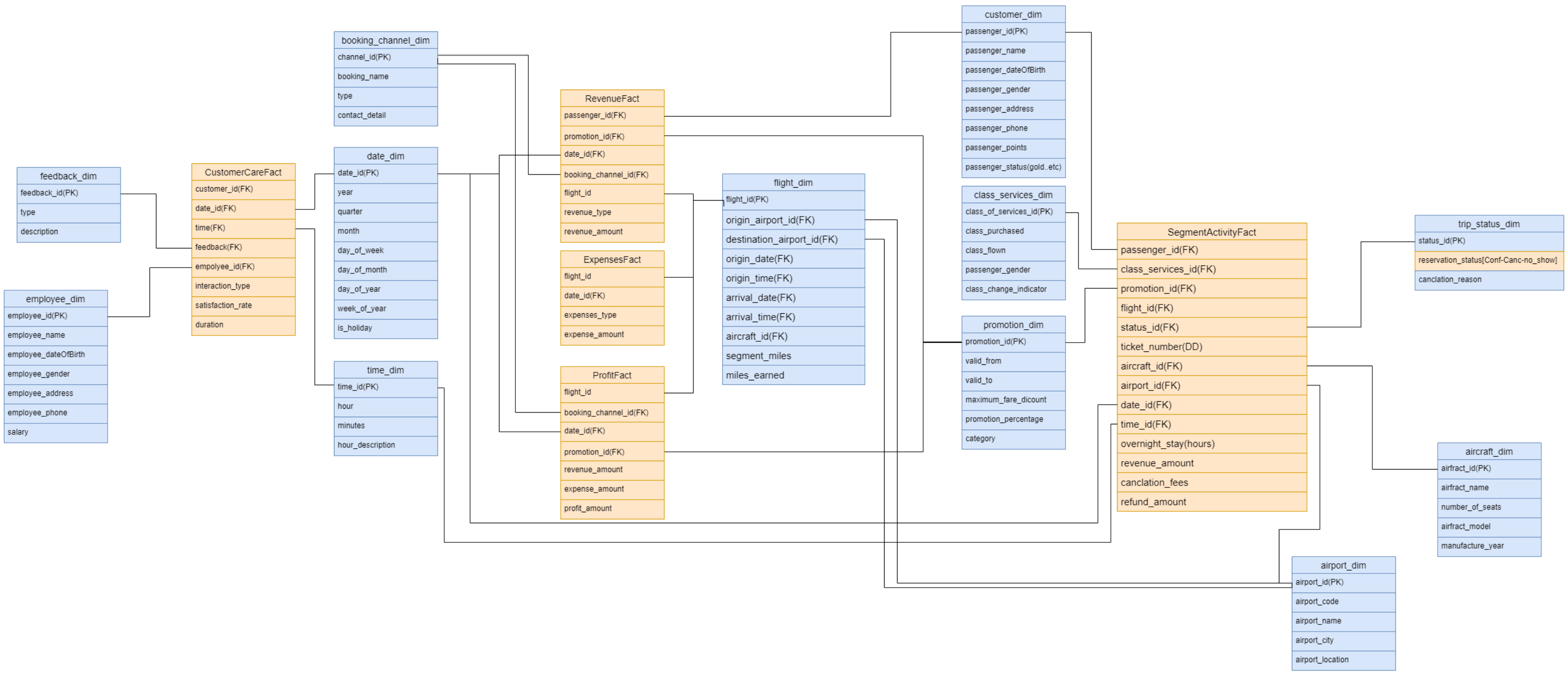
- **Business Process**: Summarizes profit-related metrics for specific flights and dates.
- **Grain**: One row per flight and date combination.
- **Fact Table**: ProfitFact
    - **Measures**:
        - **revenue_amount**: Revenue generated from the flight.
        - **expense_amount**: Expenses incurred for the flight.
        - **profit_amount**: Profit calculated as revenue minus expenses.
    - **Dimensions**:
        - **flight_id**: Links to the flight_dim table to identify the flight.
        - **date_id**: Links to the date_dim table to track the date of the flight.
        - **promotion_id**: Links to the promotion_dim table to track any promotions applied.
        - **booking_channel_id**: Links to the booking_channel_dim table to track the booking channel used.

## Customer Care

- **Business Process**: Tracks customer care interactions, including inquiries, complaints, and feedback.
- **Grain**: One row per customer care interaction.
- **Fact Table**: CustomerCareFact
    - **Measures**:

        - **satisfaction_rate**: Customer satisfaction rating for the interaction.
        - **duration**: Duration of the interaction.
    - **Dimensions**:
        - **customer_id**: Links to the customer_dim table to identify the customer.
        - **date_id**: Links to the date_dim table to track the date of the interaction.
        - **feedback_id**: Links to the feedback_dim table to track the type and description of feedback.
        - **employee_id**: Links to the employee_dim table to identify the employee handling the interaction.
        - **Interaction_type**: a degenerating dimension that refer to the way the customer contact with the company (phone call, website, etc)

# Logical Schema



## Why Choose a Dimensional Model (Star Schema)?

1. **Simplicity and Understandability**:
    - The star schema is intuitive and easy to understand for business users and analysts. It consists of a central fact table surrounded by dimension tables, making it straightforward to navigate and query.
2. **Query Performance**:
    - Star schemas are optimized for read-heavy operations, such as aggregations and joins, which are common in analytical queries. The denormalized structure reduces the number of joins required, improving query performance.
3. **Scalability**:
    - The model is scalable and can handle large volumes of data efficiently. Fact tables store transactional data, while dimension tables store descriptive attributes, allowing for efficient storage and retrieval.
4. **Alignment with Business Processes**:
    - The dimensional model aligns well with the airline's business processes, such as tracking flight activity, revenues, expenses, profits, and customer care interactions. Each fact table represents a specific business process, and the dimensions provide context for analysis.
5. **Support for Historical Data**:
    - The model supports slowly changing dimensions (SCDs), such as customer_dim, which tracks changes in customer status over time. This is critical for analyzing trends and historical data.

## What Does the Data Represent?

The data represents the airline's core business processes, including flight activity, revenue generation, expense tracking, profit analysis, and customer care interactions. Each component of the model (fact tables and dimension tables) plays a specific role in capturing and organizing this data.

**Details About Each Model Component**

## Fact Tables

Fact tables store measurable data (facts) and are the center of the star schema. They are linked to dimension tables via foreign keys.

- `SegmentActivityFact` :
  - Represents trip-related activities, such as revenue, cancellations, refunds, and overnight stays.
  - **Measures**: overnight_stay, revenue_amount, cancellation_fees, refund_amount.
  - **Dimensions**: passenger_id, class_services_id, promotion_id, flight_id, status_id, date_id, time_id.

```
-- Create SegmentActivityFact table
CREATE TABLE SegmentActivityFact (
    passenger_id NUMBER,
    class_services_id NUMBER,
    promotion_id NUMBER,
    flight_id NUMBER,
    status_id NUMBER,
    ticket_number VARCHAR2(50) PRIMARY KEY,
    overnight_stay NUMBER,
    revenue_amount NUMBER,
    cancellation_fees NUMBER,
    refund_amount NUMBER,
    date_id DATE,
    time_id TIMESTAMP,
    CONSTRAINT fk_passenger FOREIGN KEY (passenger_id) REFERENCES customer_dim(sk_passenger_id),
    CONSTRAINT fk_class_services FOREIGN KEY (class_services_id) REFERENCES class_services_dim(class_of_services_id),
    CONSTRAINT fk_promotion_sgement FOREIGN KEY (promotion_id) REFERENCES promotion_dim(promotion_id),
    CONSTRAINT fk_status FOREIGN KEY (status_id) REFERENCES trip_status_dim(status_id),
    CONSTRAINT fk_flight FOREIGN KEY (flight_id) REFERENCES flight_dim(flight_id),
    CONSTRAINT fk_origin_date_seg FOREIGN KEY (date_id) REFERENCES date_dim(date_id),
    CONSTRAINT fk_origin_time_seg FOREIGN KEY (time_id) REFERENCES time_dim(time_id)
);
```

- `RevenueFact` :
  - Tracks revenue transactions associated with passengers, flights, and dates.
  - **Measures**: revenue_amount.
  - **Dimensions**: passenger_id, date_id, flight_id, promotion_id, booking_channel_id.

```
-- Create RevenueFact table
CREATE TABLE RevenueFact (
    passenger_id NUMBER NOT NULL,
    date_id DATE NOT NULL,
    flight_id NUMBER NOT NULL,
    promotion_id NUMBER,
    booking_channel_id NUMBER,
    revenue_type VARCHAR2(255),
    revenue_amount NUMBER(15,2),
    CONSTRAINT pk_revenue PRIMARY KEY (passenger_id, date_id, flight_id, revenue_type),
    CONSTRAINT fk_promotion_rev FOREIGN KEY (promotion_id) REFERENCES promotion_dim(promotion_id),
    CONSTRAINT fk_booking_channel_rev FOREIGN KEY (booking_channel_id) REFERENCES booking_channel_dim(channel_id),
    CONSTRAINT fk_rev_passenger FOREIGN KEY (passenger_id) REFERENCES customer_dim(sk_passenger_id),
    CONSTRAINT fk_rev_date FOREIGN KEY (date_id) REFERENCES date_dim(date_id),
    CONSTRAINT fk_rev_flight FOREIGN KEY (flight_id) REFERENCES flight_dim(flight_id)
);
```

- `ExpensesFact` :
  - Tracks expense transactions associated with flights and dates.
  - **Measures**: expense_amount.
  - **Dimensions**: date_id, flight_id.

```
-- Create ExpensesFact table
CREATE TABLE ExpensesFact (
    date_id DATE NOT NULL,
    flight_id NUMBER NOT NULL,
    expenses_type VARCHAR2(255),
    expense_amount NUMBER(15,2),
    CONSTRAINT pk_expenses PRIMARY KEY (date_id, flight_id, expenses_type),
    CONSTRAINT fk_exp_date FOREIGN KEY (date_id) REFERENCES date_dim(date_id),
    CONSTRAINT fk_exp_flight FOREIGN KEY (flight_id) REFERENCES flight_dim(flight_id)
);
```

- `ProfitFact` :
  - Summarizes profit-related metrics for specific flights and dates.
  - **Measures**: revenue_amount, expense_amount, profit_amount.
  - **Dimensions**: flight_id, date_id, promotion_id, booking_channel_id.

```
-- Create ProfitFact table
CREATE TABLE ProfitFact (
    flight_id NUMBER NOT NULL,
    date_id DATE NOT NULL,
    promotion_id NUMBER,
    booking_channel_id NUMBER,
    revenue_amount NUMBER(15,2),
    expense_amount NUMBER(15,2),
    profit_amount NUMBER(15,2),
    CONSTRAINT pk_profit PRIMARY KEY (flight_id, date_id),
    CONSTRAINT fk_promotion FOREIGN KEY (promotion_id) REFERENCES promotion_dim(promotion_id),
    CONSTRAINT fk_booking_channel FOREIGN KEY (booking_channel_id) REFERENCES booking_channel_dim(channel_id),
    CONSTRAINT fk_profit_flight FOREIGN KEY (flight_id) REFERENCES flight_dim(flight_id),
    CONSTRAINT fk_profit_date FOREIGN KEY (date_id) REFERENCES date_dim(date_id)
);
```

- `CustomerCareFact` :
  - Tracks customer care interactions, including inquiries, complaints, and feedback.
  - **Measures**: satisfaction_rate, duration.
  - **Dimensions**: customer_id, date_id, feedback_id, employee_id.

```
-- Create CustomerCareFact table
CREATE TABLE CustomerCareFact (
    customer_id NUMBER,
    date_id DATE,
    time_id TIMESTAMP,
    feedback_id NUMBER,
    employee_id NUMBER,
    interaction_type VARCHAR2(50),
    satisfaction_rate NUMBER(5,2),
    duration NUMBER,
    CONSTRAINT pk_customer_care PRIMARY KEY (customer_id, date_id, feedback_id, employee_id,time_id),
    cONSTRAINT fk_customer_id FOREIGN KEY (customer_id) REFERENCES customer_dim(sk_passenger_id),
    CONSTRAINT fk_care_date FOREIGN KEY (date_id) REFERENCES date_dim(date_id),
    CONSTRAINT fk_care_feedback FOREIGN KEY (feedback_id) REFERENCES feedback_dim(feedback_id),
    CONSTRAINT fk_care_employee FOREIGN KEY (employee_id) REFERENCES employee_dim(sk_employee_id),
    CONSTRAINT fk_care_time FOREIGN KEY (time_id) REFERENCES time_dim(time_id)
);
```

### Dimension Tables

Dimension tables provide context for the facts and store descriptive attributes. They are connected to fact tables via foreign keys.

- `aircraft_dim` :
  - Stores information about aircraft, such as aircraft_name, number_of_seats, aircraft_model, and manufacture_year.
  - Used to analyze flight performance based on aircraft type.

```
-- Create aircraft_dim table
CREATE TABLE aircraft_dim (
    aircraft_id NUMBER PRIMARY KEY,
    aircraft_name VARCHAR2(100),
    number_of_seats NUMBER,
    aircraft_model VARCHAR2(50),
    manufacture_year NUMBER
);
```

- `airport_dim` :
  - Stores information about airports, such as airport_code, airport_name, airport_city, and airport_location.
  - Used to analyze flight routes and airport performance.

```
-- Create airport_dim table
CREATE TABLE airport_dim (
    airport_id NUMBER PRIMARY KEY,
    airport_code VARCHAR2(10),
    airport_name VARCHAR2(100),
    airport_city VARCHAR2(100),
    airport_location VARCHAR2(100)
);
```

- `customer_dim` :
  - Stores information about passengers, such as passenger_name, passenger_dateOfBirth, passenger_gender, passenger_address, passenger_phone, passenger_points, and passenger_status.
  - Supports analysis of customer behavior, loyalty programs, and demographics.

```
-- Create customer_dim table
CREATE TABLE customer_dim (
    sk_passenger_id NUMBER PRIMARY key,
    passenger_id NUMBER ,
    passenger_name VARCHAR2(100),
    passenger_dateOfBirth DATE,
    passenger_gender VARCHAR2(10),
    passenger_address VARCHAR2(200),
    passenger_phone VARCHAR2(15),
    passenger_points NUMBER,
    passenger_status VARCHAR2(50),
    start_date DATE DEFAULT TO_DATE('2000-01-01', 'YYYY-MM-DD'),
    end_date DATE DEFAULT TO_DATE('9999-12-31', 'YYYY-MM-DD'),
    is_current CHAR(1) DEFAULT 'Y' CHECK (is_current IN ('Y', 'N'))
);
```

- `booking_channel_dim` :
  - Stores information about booking channels, such as booking_name, type, and contact_detail.

    - Used to analyze revenue and bookings by channel (e.g., online, travel agency).

```
-- Create booking_channel_dim table
CREATE TABLE booking_channel_dim (
    channel_id NUMBER PRIMARY KEY,
    booking_name VARCHAR2(100),
    type VARCHAR2(50),
    contact_detail VARCHAR2(100)
);
```

- `trip_status_dim` :
    - Stores information about trip statuses, such as reservation_status and cancellation_reason.
    - Used to analyze trip cancellations and reservations.

```
-- Create trip_status_dim table
CREATE TABLE trip_status_dim (
    status_id NUMBER PRIMARY KEY,
    reservation_status VARCHAR2(50),
    cancellation_reason VARCHAR2(100)
);
```

- `class_services_dim` :
    - Stores information about class of services, such as class_purchased, class_flown, and class_change_indicator.
    - Used to analyze upgrades, downgrades, and class preferences.

```
-- Create class_services_dim table
CREATE TABLE class_services_dim (
    class_of_services_id NUMBER PRIMARY KEY,
    class_purchased VARCHAR2(50),
    class_flown VARCHAR2(50),
    class_change_indicator VARCHAR2(20)
);
```

- `promotion_dim` :
    - Stores information about promotions, such as valid_from, valid_to, maximum_fare_discount, promotion_percentage, and category.
    - Used to analyze the impact of promotions on revenue and bookings.

```
-- Create promotion_dim table
CREATE TABLE promotion_dim (
    promotion_id NUMBER PRIMARY KEY,
    valid_from DATE,
    valid_to DATE,
    maximum_fare_discount NUMBER,
    promotion_percentage NUMBER,
    category VARCHAR2(50)
);
```

- `time_dim` :
    - Stores time-related information, such as hour, minute, and hour_description.
    - Used to analyze flight schedules and time-based trends.

```
CREATE TABLE time_dim (
    time_id TIMESTAMP PRIMARY KEY,
    hour NUMBER(2,0) NOT NULL,      -- Stores only the hour (0-23)
    minute NUMBER(2,0) NOT NULL,    -- Stores only the minutes (0-59)
    hour_description VARCHAR2(255)  -- Descriptive text about the hour
);
```

- `date_dim` :
    - Stores date-related information, such as year, quarter, month, day_of_week, day_of_month, day_of_year, week_of_year, and is_holiday.
    - Used to analyze trends over time (e.g., seasonal patterns).

```
-- Create date_dim table
CREATE TABLE date_dim (
    date_id DATE PRIMARY KEY,
    year NUMBER(4,0),
    quarter NUMBER(1,0),
    month NUMBER(2,0),
    day_of_week NUMBER(1,0),
    day_of_month NUMBER(2,0),
    day_of_year NUMBER(3,0),
    week_of_year NUMBER(2,0),
    is_holiday NUMBER(1,0) CHECK (is_holiday IN (0,1))
);
```

- `flight_dim` :
    - Stores information about flights, such as origin_airport_id, destination_airport_id, origin_date, origin_time, arrival_date, arrival_time, aircraft_id, segment_miles, and miles_earned.
    - Used to analyze flight performance, routes, and mileage.

```
-- Create flight_dim table
CREATE TABLE flight_dim (
    flight_id NUMBER PRIMARY KEY,
    origin_airport_id NUMBER NOT NULL,
    destination_airport_id NUMBER NOT NULL,
    origin_date DATE NOT NULL,
    origin_time TIMESTAMP NOT NULL,
    arrival_date DATE NOT NULL,
    arrival_time TIMESTAMP NOT NULL,
    aircraft_id NUMBER NOT NULL,
    segment_miles NUMBER(10,2),
    miles_earned NUMBER(10,2),
    CONSTRAINT fk_origin_airport FOREIGN KEY (origin_airport_id) REFERENCES airport_dim(airport_id),
    CONSTRAINT fk_destination_airport FOREIGN KEY (destination_airport_id) REFERENCES airport_dim(airport_id),
    CONSTRAINT fk_origin_date FOREIGN KEY (origin_date) REFERENCES date_dim(date_id),
    CONSTRAINT fk_origin_time FOREIGN KEY (origin_time) REFERENCES time_dim(time_id),
    CONSTRAINT fk_arrival_date FOREIGN KEY (arrival_date) REFERENCES date_dim(date_id),
    CONSTRAINT fk_arrival_time FOREIGN KEY (arrival_time) REFERENCES time_dim(time_id),
    CONSTRAINT fk_aircraft FOREIGN KEY (aircraft_id) REFERENCES aircraft_dim(aircraft_id)
);
```

- `feedback_dim` :
    - Stores information about customer feedback, such as type and description.
    - Used to analyze customer satisfaction and issues.

```
-- Create feedback_dim table
CREATE TABLE feedback_dim (
    feedback_id NUMBER PRIMARY KEY,
    type VARCHAR2(50),
    description VARCHAR2(500)
);
```

- `employee_dim` :
    - Stores information about employees, such as employee_name, employee_dateOfBirth, employee_gender, employee_address, employee_phone, and salary.
    - Used to analyze employee performance and customer care interactions.

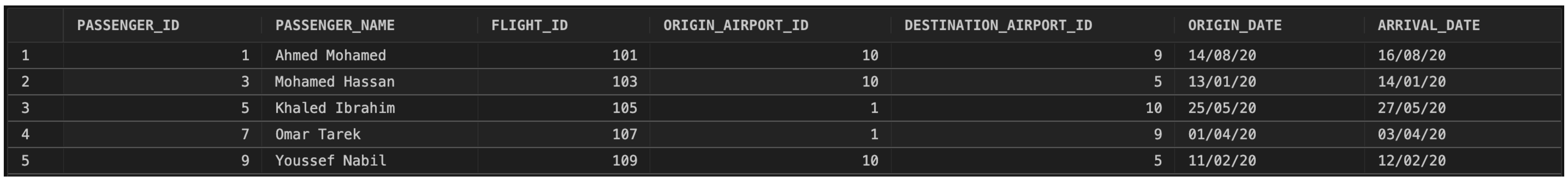
```
-- Create employee_dim table
CREATE TABLE employee_dim (
  sk_employee_id NUMBER PRIMARY KEY,
  employee_id NUMBER,
  employee_name VARCHAR2(35),
  employee_dateOfBirth DATE,
  employee_gender VARCHAR2(10),
  employee_address VARCHAR2(100),
  employee_phone VARCHAR2(20),
  salary NUMBER(10,2),
  start_date DATE DEFAULT SYSDATE,
  end_date DATE,
  is_current CHAR(1) DEFAULT 'Y' CHECK (is_current IN ('Y', 'N'))
);
```

## Why This Design Works for our case?

1. **Business Process Alignment**:
    - Each fact table corresponds to a specific business process (e.g., flight activity, revenue, expenses, profit, customer care), ensuring that the model supports the airline's analytical needs.
2. **Granularity**:
    - The grain of each fact table is carefully chosen to capture the required level of detail. For example, SegmentActivityFact captures trip-level details, while RevenueFact captures transaction-level details.
3. **Flexibility**:
    - The model is flexible and can be extended to include additional dimensions or measures as the airline's business evolves.
4. **Historical Analysis**:
    - Slowly changing dimensions (e.g., customer_dim) allow the airline to track changes over time, such as customer status or employee roles.
5. **Performance**:
    - The star schema's denormalized structure ensures fast query performance, which is critical for large-scale analytical workloads.

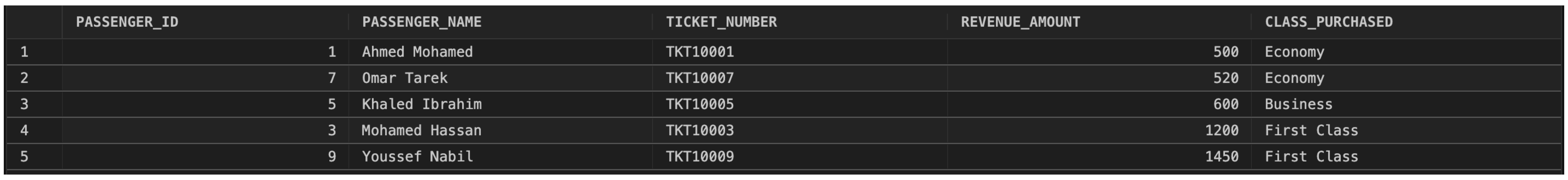
## Queries support decision making

```
--1. Flights the company’s frequent flyers take
SELECT c.passenger_id, c.passenger_name, f.flight_id, f.origin_airport_id, f.destination_airport_id, f.origin_date, f.arrival_date
FROM SegmentActivityFact s
JOIN customer_dim c ON s.passenger_id = c.sk_passenger_id
JOIN flight_dim f ON s.flight_id = f.flight_id
WHERE c.passenger_status IN ('Gold', 'Platinum', 'Aluminum', 'Titanium');
```



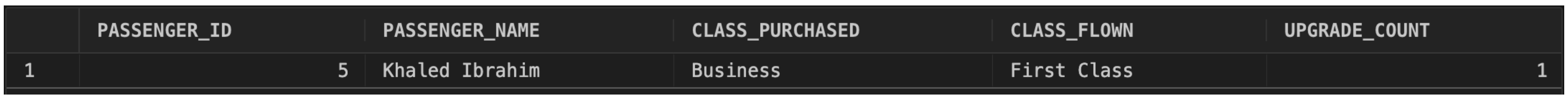
| | PASSENGER_ID | PASSENGER_NAME | FLIGHT_ID | ORIGIN_AIRPORT_ID | DESTINATION_AIRPORT_ID | ORIGIN_DATE | ARRIVAL_DATE |
|---|---|---|---|---|---|---|---|
| 1 | 1 | Ahmed Mohamed | 101 | 10 | 9 | 14/08/20 | 16/08/20 |
| 2 | 3 | Mohamed Hassan | 103 | 10 | 5 | 13/01/20 | 14/01/20 |
| 3 | 5 | Khaled Ibrahim | 105 | 1 | 10 | 25/05/20 | 27/05/20 |
| 4 | 7 | Omar Tarek | 107 | 1 | 9 | 01/04/20 | 03/04/20 |
| 5 | 9 | Youssef Nabil | 109 | 10 | 5 | 11/02/20 | 12/02/20 |

```
--2. What fare basis they pay
SELECT s.passenger_id, c.passenger_name, s.ticket_number, s.revenue_amount, cs.class_purchased
FROM SegmentActivityFact s
JOIN customer_dim c ON s.passenger_id = c.sk_passenger_id
JOIN class_services_dim cs ON s.class_services_id = cs.class_of_services_id
WHERE c.passenger_status IN ('Gold', 'Platinum', 'Aluminum', 'Titanium');
```



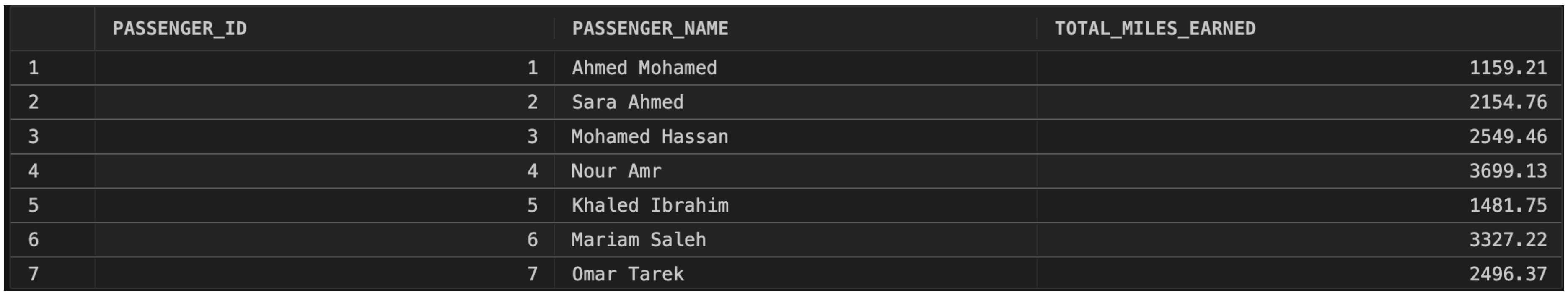
| | PASSENGER_ID | PASSENGER_NAME | TICKET_NUMBER | REVENUE_AMOUNT | CLASS_PURCHASED |
|---|---|---|---|---|---|
| 1 | 1 | Ahmed Mohamed | TKT10001 | 500 | Economy |
| 2 | 7 | Omar Tarek | TKT10007 | 520 | Economy |
| 3 | 5 | Khaled Ibrahim | TKT10005 | 600 | Business |
| 4 | 3 | Mohamed Hassan | TKT10003 | 1200 | First Class |
| 5 | 9 | Youssef Nabil | TKT10009 | 1450 | First Class |

```
--3. How often they upgrade
SELECT c.passenger_id, c.passenger_name, cs.class_purchased, cs.class_flown, COUNT(*) AS upgrade_count
FROM SegmentActivityFact s
JOIN customer_dim c ON s.passenger_id = c.sk_passenger_id
JOIN class_services_dim cs ON s.class_services_id = cs.class_of_services_id
--WHERE cs.class_purchased <> cs.class_flown
WHERE cs.class_change_indicator = 'Upgrade'
AND c.passenger_status IN ('Gold', 'Platinum', 'Aluminum', 'Titanium')
GROUP BY c.passenger_id, c.passenger_name, cs.class_purchased, cs.class_flown;
```

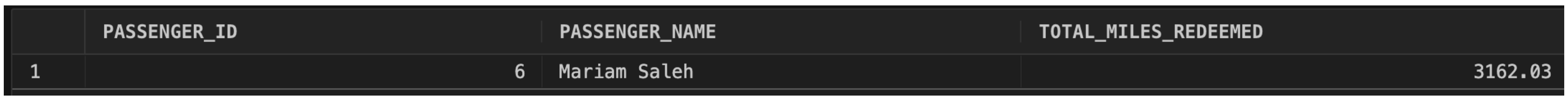


| | PASSENGER_ID | PASSENGER_NAME | CLASS_PURCHASED | CLASS_FLOWN | UPGRADE_COUNT |
|---|---|---|---|---|---|
| 1 | 5 | Khaled Ibrahim | Business | First Class | 1 |

```
--4. How they earn and redeem their frequent flyer miles
SELECT c.passenger_id, c.passenger_name, SUM(f.miles_earned) AS total_miles_earned
FROM SegmentActivityFact s
JOIN customer_dim c ON s.passenger_id = c.sk_passenger_id
JOIN flight_dim f ON s.flight_id = f.flight_id
GROUP BY c.passenger_id, c.passenger_name;
------------------
SELECT
    s.passenger_id,
    c.passenger_name,
    f.segment_miles AS total_miles_redeemed
FROM SegmentActivityFact s
JOIN customer_dim c ON s.passenger_id = c.sk_passenger_id
JOIN flight_dim f ON s.flight_id = f.flight_id
AND s.revenue_amount = 0  -- zero revenue on a flight means that the passenger use his earned miles and redeem his points
```



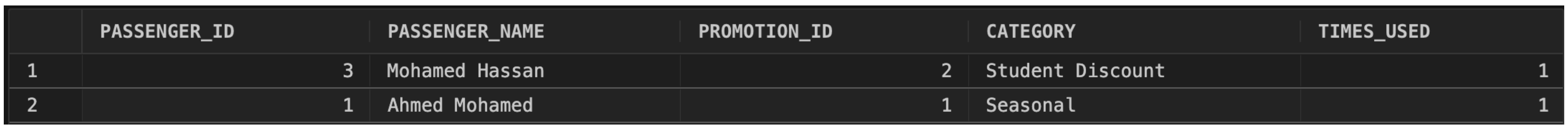
| | PASSENGER_ID | PASSENGER_NAME | TOTAL_MILES_EARNED |
|---|---|---|---|
| 1 | 1 | Ahmed Mohamed | 1159.21 |
| 2 | 2 | Sara Ahmed | 2154.76 |
| 3 | 3 | Mohamed Hassan | 2549.46 |
| 4 | 4 | Nour Amr | 3699.13 |
| 5 | 5 | Khaled Ibrahim | 1481.75 |
| 6 | 6 | Mariam Saleh | 3327.22 |
| 7 | 7 | Omar Tarek | 2496.37 |



| | PASSENGER_ID | PASSENGER_NAME | TOTAL_MILES_REDEEMED |
|---|---|---|---|
| 1 | 6 | Mariam Saleh | 3162.03 |

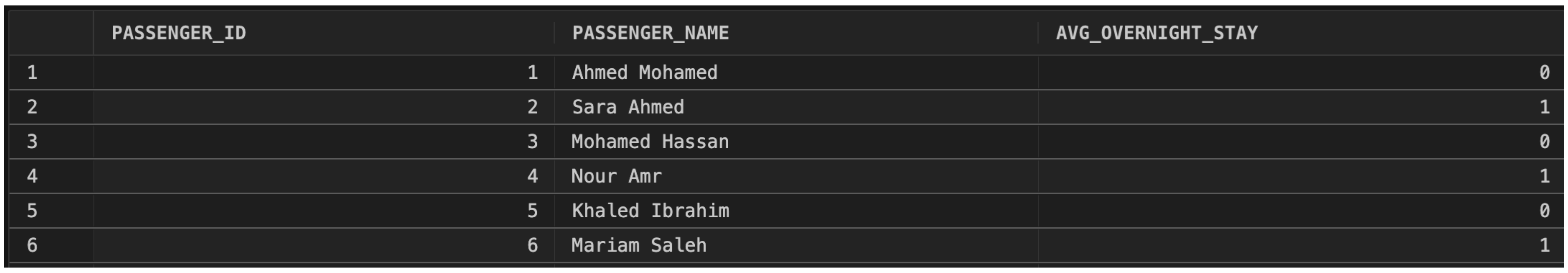
```
--5. Whether they respond to special fare promotions
SELECT c.passenger_id, c.passenger_name, p.promotion_id, p.category, COUNT(*) AS times_used
FROM SegmentActivityFact s
JOIN customer_dim c ON s.passenger_id = c.sk_passenger_id
JOIN promotion_dim p ON s.promotion_id = p.promotion_id
WHERE c.passenger_status IN ('Gold', 'Platinum', 'Aluminum', 'Titanium')
```

```
GROUP BY c.passenger_id, c.passenger_name, p.promotion_id, p.category
ORDER BY times_used DESC;
```



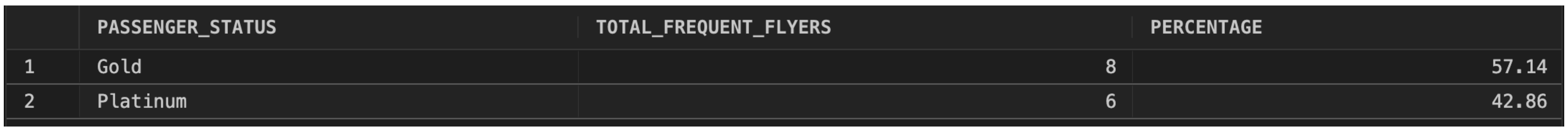
| | PASSENGER_ID | PASSENGER_NAME | PROMOTION_ID | CATEGORY | TIMES_USED |
|---|---|---|---|---|---|
| 1 | 3 | Mohamed Hassan | 2 | Student Discount | 1 |
| 2 | 1 | Ahmed Mohamed | 1 | Seasonal | 1 |

```
--6. How long their overnight stays are
SELECT c.passenger_id, c.passenger_name, AVG(s.overnight_stay) AS avg_overnight_stay
FROM SegmentActivityFact s
JOIN customer_dim c ON s.passenger_id = c.sk_passenger_id
GROUP BY c.passenger_id, c.passenger_name;
```



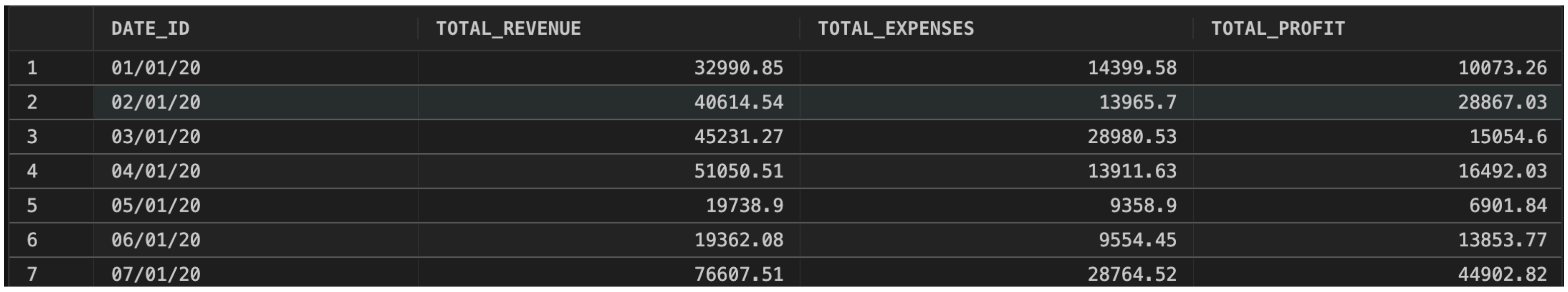
| | PASSENGER_ID | PASSENGER_NAME | AVG_OVERNIGHT_STAY |
|---|---|---|---|
| 1 | 1 | Ahmed Mohamed | 0 |
| 2 | 2 | Sara Ahmed | 1 |
| 3 | 3 | Mohamed Hassan | 0 |
| 4 | 4 | Nour Amr | 1 |
| 5 | 5 | Khaled Ibrahim | 0 |
| 6 | 6 | Mariam Saleh | 1 |

```
--7. Proportion of frequent flyers by status
SELECT passenger_status, COUNT(*) AS total_frequent_flyers,
    ROUND(100.0 * COUNT(*) / (SELECT COUNT(*) FROM customer_dim WHERE passenger_status IN ('Gold', 'Platinum', 'Aluminum', 'Titanium')), 2) AS percentage
FROM customer_dim
WHERE passenger_status IN ('Gold', 'Platinum', 'Aluminum', 'Titanium')
GROUP BY passenger_status;
```



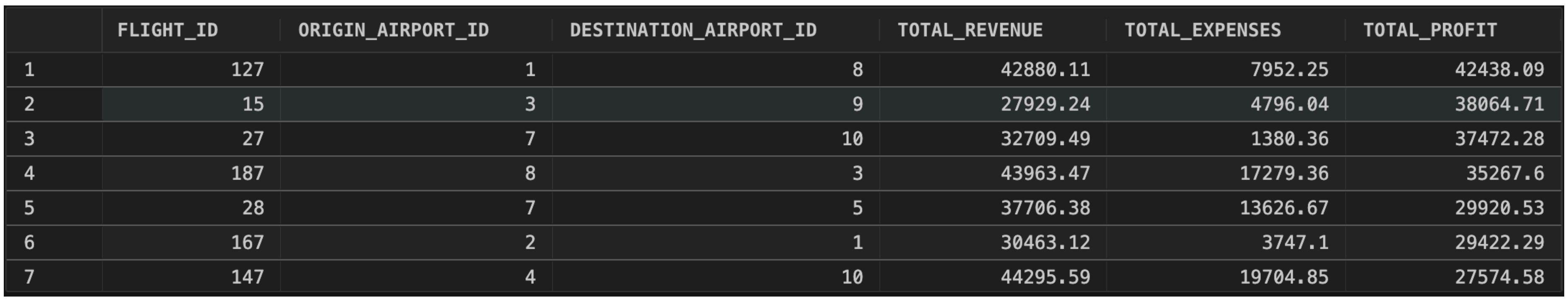
| | PASSENGER_STATUS | TOTAL_FREQUENT_FLYERS | PERCENTAGE |
|---|---|---|---|
| 1 | Gold | 8 | 57.14 |
| 2 | Platinum | 6 | 42.86 |

```
--8.total profit for each date to analyze profit trends over time.
SELECT
    date_id,
    SUM(revenue_amount) AS total_revenue,
    SUM(expense_amount) AS total_expenses,
    SUM(profit_amount) AS total_profit
FROM ProfitFact
GROUP BY date_id
ORDER BY date_id;
```



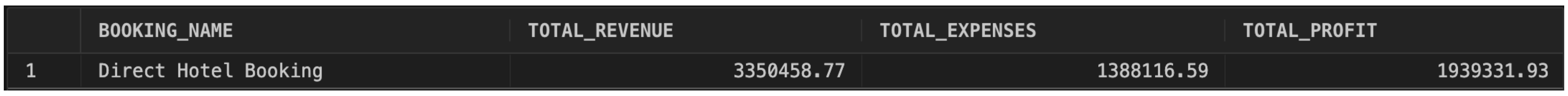
| | DATE_ID | TOTAL_REVENUE | TOTAL_EXPENSES | TOTAL_PROFIT |
|---|---|---|---|---|
| 1 | 01/01/20 | 32990.85 | 14399.58 | 10073.26 |
| 2 | 02/01/20 | 40614.54 | 13965.7 | 28867.03 |
| 3 | 03/01/20 | 45231.27 | 28980.53 | 15054.6 |
| 4 | 04/01/20 | 51050.51 | 13911.63 | 16492.03 |
| 5 | 05/01/20 | 19738.9 | 9358.9 | 6901.84 |
| 6 | 06/01/20 | 19362.08 | 9554.45 | 13853.77 |
| 7 | 07/01/20 | 76607.51 | 28764.52 | 44902.82 |

```
--9.profitability of each flight.
SELECT
    pf.flight_id,
    f.origin_airport_id,
    f.destination_airport_id,
    SUM(pf.revenue_amount) AS total_revenue,
    SUM(pf.expense_amount) AS total_expenses,
    SUM(pf.profit_amount) AS total_profit
FROM ProfitFact pf
JOIN flight_dim f ON pf.flight_id = f.flight_id
GROUP BY pf.flight_id, f.origin_airport_id, f.destination_airport_id
ORDER BY total_profit DESC;
```



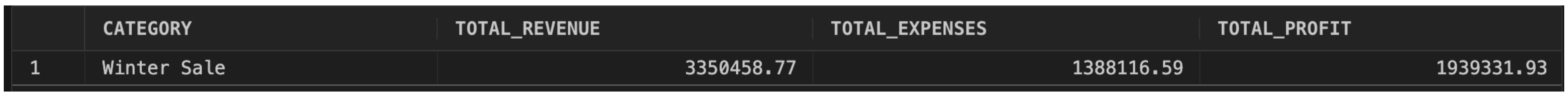
| | FLIGHT_ID | ORIGIN_AIRPORT_ID | DESTINATION_AIRPORT_ID | TOTAL_REVENUE | TOTAL_EXPENSES | TOTAL_PROFIT |
|---|---|---|---|---|---|---|
| 1 | 127 | 1 | 8 | 42880.11 | 7952.25 | 42438.09 |
| 2 | 15 | 3 | 9 | 27929.24 | 4796.04 | 38064.71 |
| 3 | 27 | 7 | 10 | 32709.49 | 1380.36 | 37472.28 |
| 4 | 187 | 8 | 3 | 43963.47 | 17279.36 | 35267.6 |
| 5 | 28 | 7 | 5 | 37706.38 | 13626.67 | 29920.53 |
| 6 | 167 | 2 | 1 | 30463.12 | 3747.1 | 29422.29 |
| 7 | 147 | 4 | 10 | 44295.59 | 19704.85 | 27574.58 |

```
--10.which booking channels generate the most profit.
SELECT
    bc.booking_name,
    SUM(pf.revenue_amount) AS total_revenue,
    SUM(pf.expense_amount) AS total_expenses,
    SUM(pf.profit_amount) AS total_profit
FROM ProfitFact pf
JOIN booking_channel_dim bc ON pf.booking_channel_id = bc.channel_id
GROUP BY bc.booking_name
ORDER BY total_profit DESC;
```



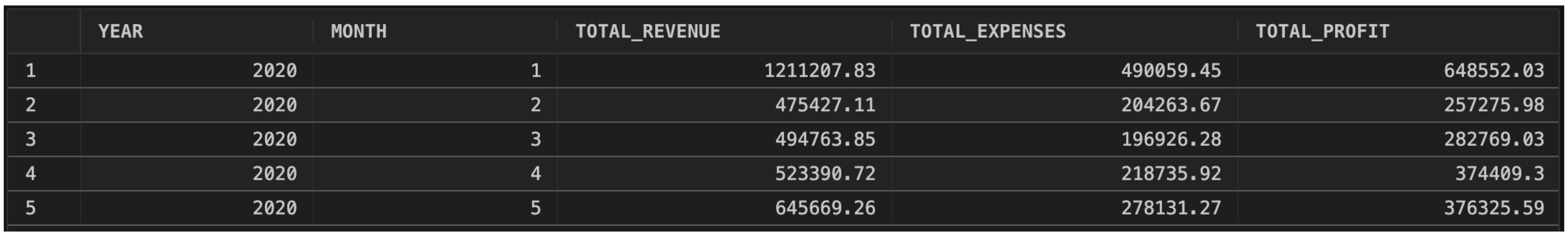
| | BOOKING_NAME | TOTAL_REVENUE | TOTAL_EXPENSES | TOTAL_PROFIT |
|---|---|---|---|---|
| 1 | Direct Hotel Booking | 3350458.77 | 1388116.59 | 1939331.93 |

```
--11.effectiveness of promotions by evaluating their contribution to profit.
SELECT
    p.category,
    SUM(pf.revenue_amount) AS total_revenue,
    SUM(pf.expense_amount) AS total_expenses,
    SUM(pf.profit_amount) AS total_profit
FROM ProfitFact pf
JOIN promotion_dim p ON pf.promotion_id = p.promotion_id
GROUP BY p.category
ORDER BY total_profit DESC;
```



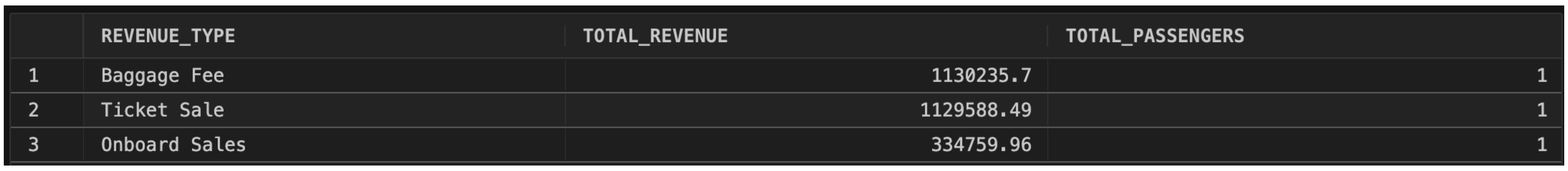
| | CATEGORY | TOTAL_REVENUE | TOTAL_EXPENSES | TOTAL_PROFIT |
|---|---|---|---|---|
| 1 | Winter Sale | 3350458.77 | 1388116.59 | 1939331.93 |

```
--12.Monthly Profit Analysis
SELECT
    d.year,
    d.month,
    SUM(pf.revenue_amount) AS total_revenue,
    SUM(pf.expense_amount) AS total_expenses,
    SUM(pf.profit_amount) AS total_profit
FROM ProfitFact pf
JOIN date_dim d ON pf.date_id = d.date_id
GROUP BY d.year, d.month
ORDER BY d.year, d.month;
```



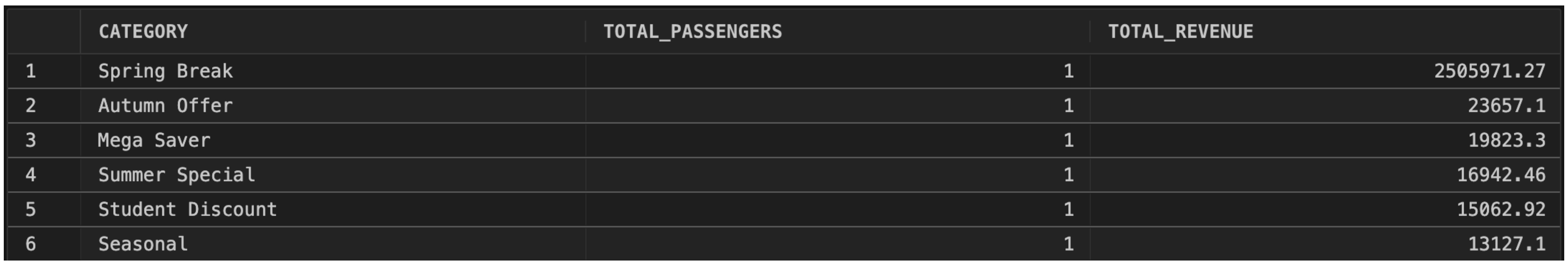
| | YEAR | MONTH | TOTAL_REVENUE | TOTAL_EXPENSES | TOTAL_PROFIT |
|---|---|---|---|---|---|
| 1 | 2020 | 1 | 1211207.83 | 490059.45 | 648552.03 |
| 2 | 2020 | 2 | 475427.11 | 204263.67 | 257275.98 |
| 3 | 2020 | 3 | 494763.85 | 196926.28 | 282769.03 |
| 4 | 2020 | 4 | 523390.72 | 218735.92 | 374409.3 |
| 5 | 2020 | 5 | 645669.26 | 278131.27 | 376325.59 |

```
--13.Revenue Distribution by Passenger Type
SELECT
    revenue_type,
    SUM(revenue_amount) AS total_revenue,
    COUNT(DISTINCT passenger_id) AS total_passengers
FROM RevenueFact
GROUP BY revenue_type
ORDER BY total_revenue DESC;
```



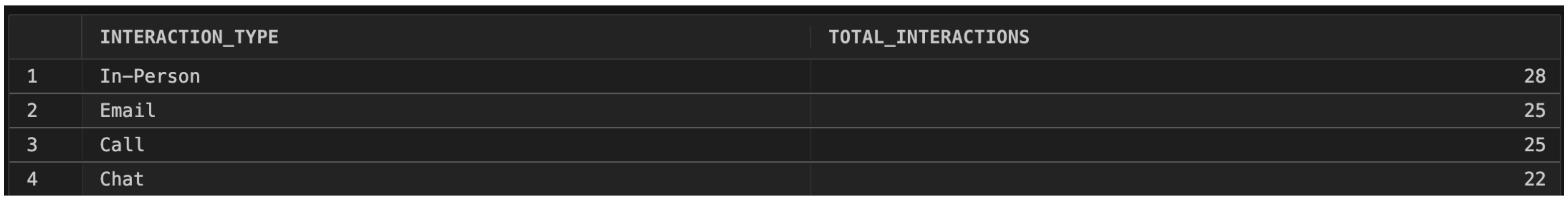
| | REVENUE_TYPE | TOTAL_REVENUE | TOTAL_PASSENGERS |
|---|---|---|---|
| 1 | Baggage Fee | 1130235.7 | 1 |
| 2 | Ticket Sale | 1129588.49 | 1 |
| 3 | Onboard Sales | 334759.96 | 1 |

```
--14.Impact of Promotions on Revenue
SELECT
    p.category,
    COUNT(DISTINCT r.passenger_id) AS total_passengers,
    SUM(r.revenue_amount) AS total_revenue
FROM RevenueFact r
JOIN promotion_dim p ON r.promotion_id = p.promotion_id
GROUP BY p.category
ORDER BY total_revenue DESC;
```



| | CATEGORY | TOTAL_PASSENGERS | TOTAL_REVENUE |
|---|---|---|---|
| 1 | Spring Break | 1 | 2505971.27 |
| 2 | Autumn Offer | 1 | 23657.1 |
| 3 | Mega Saver | 1 | 19823.3 |
| 4 | Summer Special | 1 | 16942.46 |
| 5 | Student Discount | 1 | 15062.92 |
| 6 | Seasonal | 1 | 13127.1 |

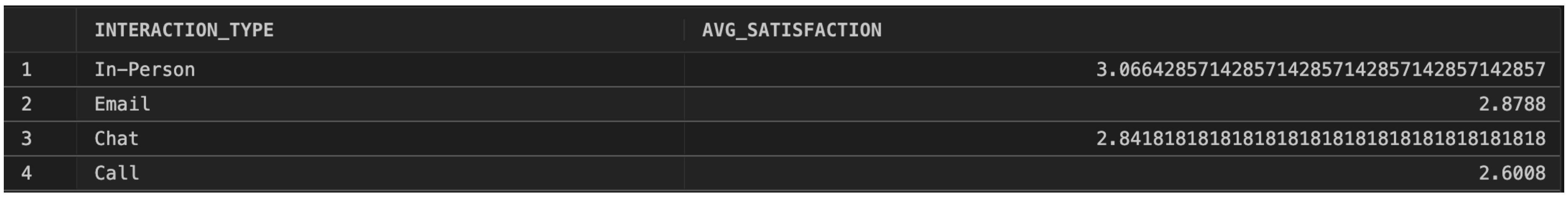
```
--15.Most Frequent Interaction Types
SELECT
    interaction_type,
    COUNT(*) AS total_interactions
FROM CustomerCareFact
GROUP BY interaction_type
ORDER BY total_interactions DESC;
```



| | INTERACTION_TYPE | TOTAL_INTERACTIONS |
|---|---|---|
| 1 | In-Person | 28 |
| 2 | Email | 25 |
| 3 | Call | 25 |
| 4 | Chat | 22 |

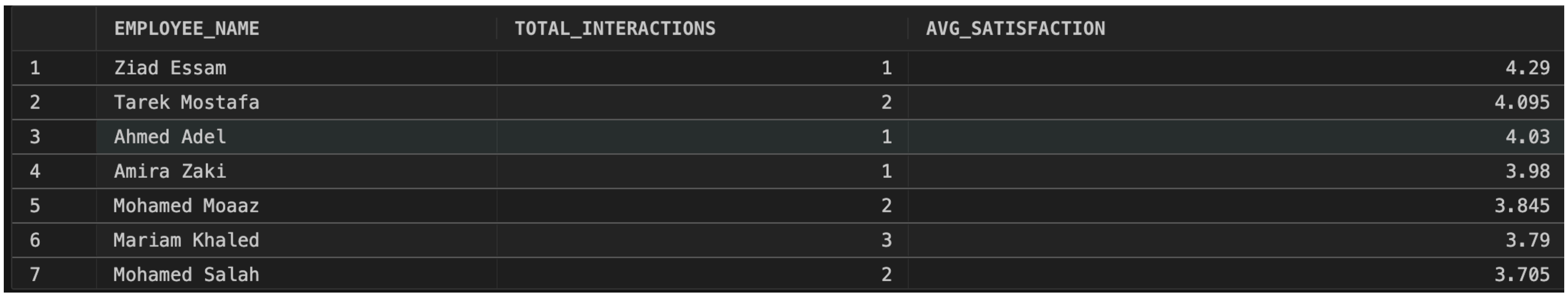
```
--16.Average Satisfaction Rate by Interaction Type
SELECT
    interaction_type,
    AVG(satisfaction_rate) AS avg_satisfaction
```

```
FROM CustomerCareFact
WHERE satisfaction_rate IS NOT NULL
GROUP BY interaction_type
ORDER BY avg_satisfaction DESC;
```



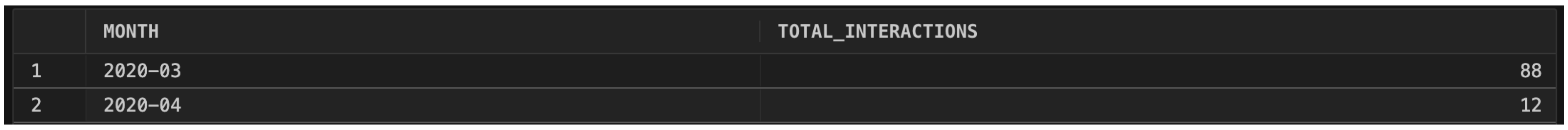
| | INTERACTION_TYPE | AVG_SATISFACTION |
|---|---|---|
| 1 | In-Person | 3.06642857142857142857142857142857142857 |
| 2 | Email | 2.8788 |
| 3 | Chat | 2.84181818181818181818181818181818181818 |
| 4 | Call | 2.6008 |

```
--17.Employee Performance in Handling Customer Care
SELECT
    e.employee_name,
    COUNT(*) AS total_interactions,
    AVG(ccf.satisfaction_rate) AS avg_satisfaction
FROM CustomerCareFact ccf
JOIN employee_dim e ON ccf.employee_id = e.sk_employee_id
GROUP BY e.employee_name
ORDER BY avg_satisfaction DESC;
```



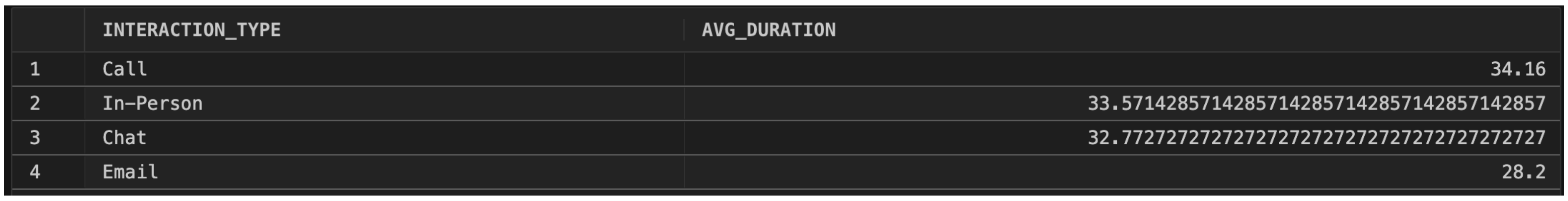
| | EMPLOYEE_NAME | TOTAL_INTERACTIONS | AVG_SATISFACTION |
|---|---|---|---|
| 1 | Ziad Essam | 1 | 4.29 |
| 2 | Tarek Mostafa | 2 | 4.095 |
| 3 | Ahmed Adel | 1 | 4.03 |
| 4 | Amira Zaki | 1 | 3.98 |
| 5 | Mohamed Moaaz | 2 | 3.845 |
| 6 | Mariam Khaled | 3 | 3.79 |
| 7 | Mohamed Salah | 2 | 3.705 |

```
--18.Customer Care Trends Over Time
SELECT
    TO_CHAR(date_id, 'YYYY-MM') AS month,
    COUNT(*) AS total_interactions
FROM CustomerCareFact
GROUP BY TO_CHAR(date_id, 'YYYY-MM')
ORDER BY month ASC;
```



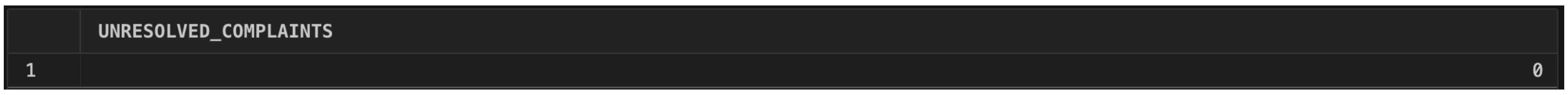
| | MONTH | TOTAL_INTERACTIONS |
|---|---|---|
| 1 | 2020-03 | 88 |
| 2 | 2020-04 | 12 |

```
--19.Average duration of interactions per type.
SELECT
    interaction_type,
    AVG(duration) AS avg_duration
FROM CustomerCareFact
WHERE duration IS NOT NULL
GROUP BY interaction_type
ORDER BY avg_duration DESC;
```



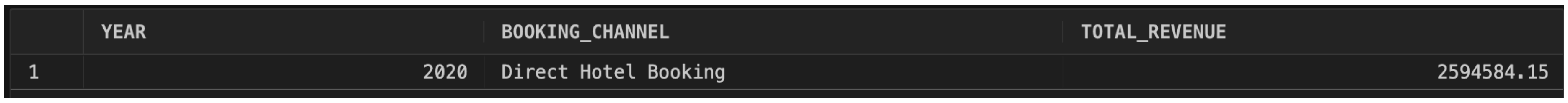
| | INTERACTION_TYPE | AVG_DURATION |
|---|---|---|
| 1 | Call | 34.16 |
| 2 | In-Person | 33.57142857142857142857142857142857142857 |
| 3 | Chat | 32.77272727272727272727272727272727272727 |
| 4 | Email | 28.2 |

```
--20.Unresolved Complaints
SELECT COUNT(*) AS unresolved_complaints
FROM CustomerCareFact
WHERE interaction_type = 'Complaint' AND feedback_id IS NULL;
```



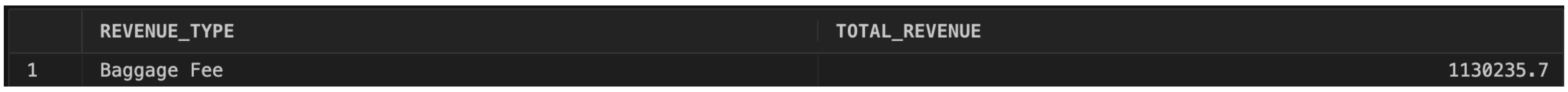
|   | UNRESOLVED_COMPLAINTS |
|---|---|
| 1 | 0 |

```
--21.top booking channel in terms of revenue by year
SELECT year, booking_channel, total_revenue
FROM (
    SELECT
        dd.year,
        bcd.booking_name AS booking_channel,
        SUM(rf.revenue_amount) AS total_revenue,
        ROW_NUMBER() OVER (PARTITION BY dd.year ORDER BY SUM(rf.revenue_amount) DESC) AS rn
    FROM RevenueFact rf
    JOIN booking_channel_dim bcd
        ON rf.booking_channel_id = bcd.channel_id
    JOIN date_dim dd
        ON rf.date_id = dd.date_id
    WHERE rf.revenue_amount IS NOT NULL
    GROUP BY dd.year, bcd.booking_name
)
WHERE rn = 1;
```



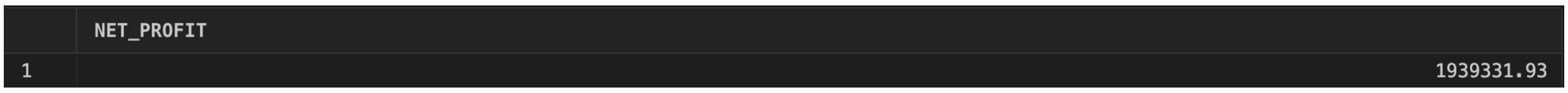
|   | YEAR | BOOKING_CHANNEL | TOTAL_REVENUE |
|---|---|---|---|
| 1 | 2020 | Direct Hotel Booking | 2594584.15 |

```
--22.top revenue type
SELECT revenue_type, total_revenue
FROM (
    SELECT
        rf.revenue_type,
        SUM(rf.revenue_amount) AS total_revenue,
        RANK() OVER (ORDER BY SUM(rf.revenue_amount) DESC) AS rnk
    FROM RevenueFact rf
    WHERE rf.revenue_amount IS NOT NULL
    GROUP BY rf.revenue_type
)
WHERE rnk = 1;
```



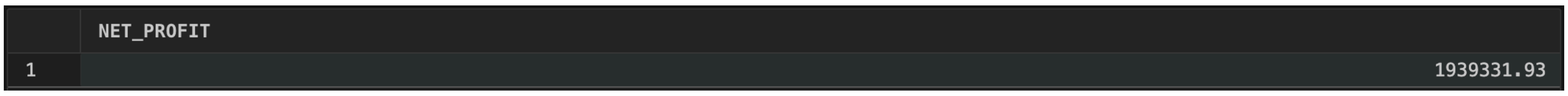
|   | REVENUE_TYPE | TOTAL_REVENUE |
|---|---|---|
| 1 | Baggage Fee | 1130235.7 |

```
--23. net profit by year
SELECT SUM(pf.profit_amount) AS net_profit
FROM ProfitFact pf
JOIN date_dim dd
    ON pf.date_id = dd.date_id
WHERE dd.year = 2020;
```



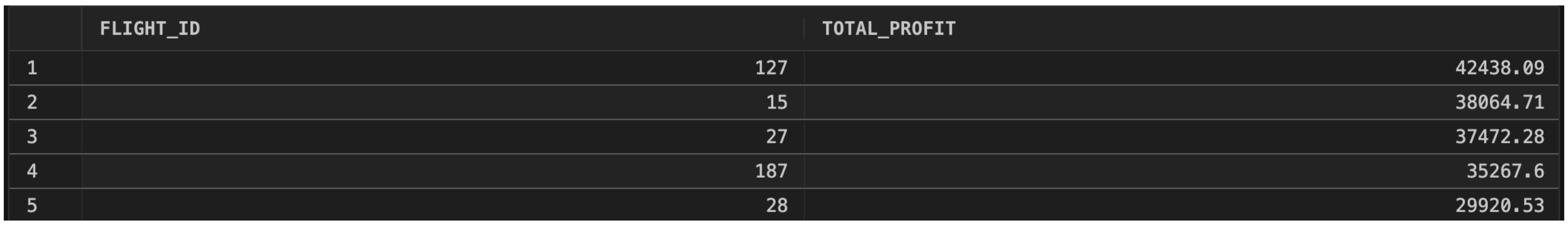
|   | NET_PROFIT |
|---|---|
| 1 | 1939331.93 |

```
--24. net profit by month by year
SELECT SUM(pf.profit_amount) AS net_profit
FROM ProfitFact pf
JOIN date_dim dd
    ON pf.date_id = dd.date_id
WHERE dd.year = 2020
AND dd.month = 1;
```



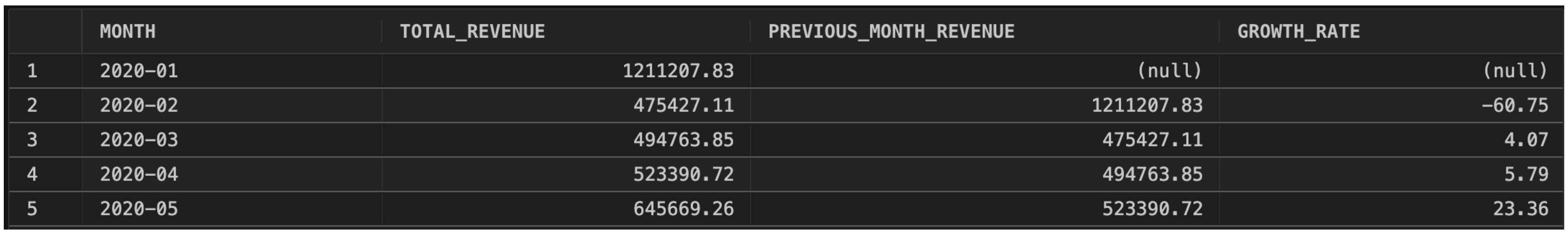
| | NET_PROFIT |
|---|---|
| 1 | 1939331.93 |

```
--Top 5 Most Profitable Flights
SELECT
    pf.flight_id,
    SUM(pf.profit_amount) AS total_profit
FROM ProfitFact pf
GROUP BY pf.flight_id
ORDER BY total_profit DESC
FETCH FIRST 5 ROWS ONLY;
```



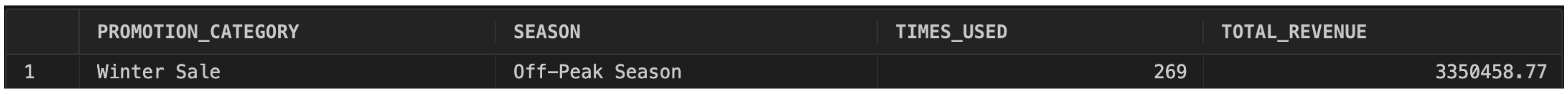
| | FLIGHT_ID | TOTAL_PROFIT |
|---|---|---|
| 1 | 127 | 42438.09 |
| 2 | 15 | 38064.71 |
| 3 | 27 | 37472.28 |
| 4 | 187 | 35267.6 |
| 5 | 28 | 29920.53 |

```
--Monthly Revenue Growth Rate
SELECT
    TO_CHAR(pf.date_id, 'YYYY-MM') AS month,
    SUM(pf.revenue_amount) AS total_revenue,
    LAG(SUM(pf.revenue_amount)) OVER (ORDER BY TO_CHAR(pf.date_id, 'YYYY-MM')) AS previous_month_revenue,
    ROUND((SUM(pf.revenue_amount) - LAG(SUM(pf.revenue_amount)) OVER (ORDER BY TO_CHAR(pf.date_id, 'YYYY-MM')))
    / NULLIF(LAG(SUM(pf.revenue_amount)) OVER (ORDER BY TO_CHAR(pf.date_id, 'YYYY-MM')), 0) * 100, 2) AS growth_rate
FROM ProfitFact pf
GROUP BY TO_CHAR(pf.date_id, 'YYYY-MM')
ORDER BY month;
```



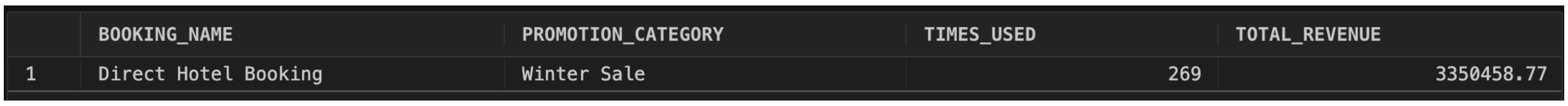
| | MONTH | TOTAL_REVENUE | PREVIOUS_MONTH_REVENUE | GROWTH_RATE |
|---|---|---|---|---|
| 1 | 2020-01 | 1211207.83 | (null) | (null) |
| 2 | 2020-02 | 475427.11 | 1211207.83 | -60.75 |
| 3 | 2020-03 | 494763.85 | 475427.11 | 4.07 |
| 4 | 2020-04 | 523390.72 | 494763.85 | 5.79 |
| 5 | 2020-05 | 645669.26 | 523390.72 | 23.36 |

```
--Best-Performing Promotions in Peak vs Off-Peak Seasons
SELECT
    pd.category AS promotion_category,
    CASE
        WHEN TO_CHAR(pf.date_id, 'MM') IN ('06', '07', '08', '12') THEN 'Peak Season'
        ELSE 'Off-Peak Season'
    END AS season,
    COUNT(pf.promotion_id) AS times_used,
    SUM(pf.revenue_amount) AS total_revenue
FROM ProfitFact pf
JOIN promotion_dim pd ON pf.promotion_id = pd.promotion_id
GROUP BY pd.category,
    CASE
        WHEN TO_CHAR(pf.date_id, 'MM') IN ('06', '07', '08', '12') THEN 'Peak Season'
        ELSE 'Off-Peak Season'
    END
ORDER BY season, total_revenue DESC;
```



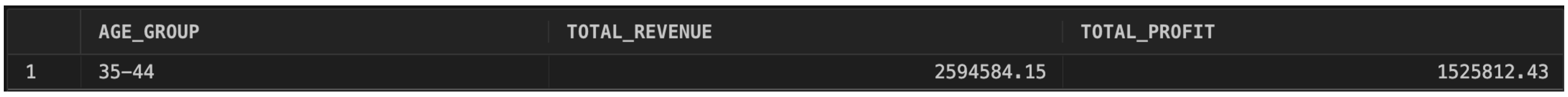
| | PROMOTION_CATEGORY | SEASON | TIMES_USED | TOTAL_REVENUE |
|---|---|---|---|---|
| 1 | Winter Sale | Off-Peak Season | 269 | 3350458.77 |

```
--Effectiveness of Promotions by Booking Channel
SELECT
    bc.booking_name,
    pd.category AS promotion_category,
    COUNT(pf.promotion_id) AS times_used,
    SUM(pf.revenue_amount) AS total_revenue
FROM ProfitFact pf
JOIN booking_channel_dim bc ON pf.booking_channel_id = bc.channel_id
JOIN promotion_dim pd ON pf.promotion_id = pd.promotion_id
GROUP BY bc.booking_name, pd.category
ORDER BY bc.booking_name, total_revenue DESC;
```



| | BOOKING_NAME | PROMOTION_CATEGORY | TIMES_USED | TOTAL_REVENUE |
|---|---|---|---|---|
| 1 | Direct Hotel Booking | Winter Sale | 269 | 3350458.77 |

```
--Most Revenue and profit according to the Age Group
SELECT
    CASE
        WHEN EXTRACT(YEAR FROM rf.date_id) - EXTRACT(YEAR FROM cd.passenger_dateofbirth) < 25 THEN '18-24'
        WHEN EXTRACT(YEAR FROM rf.date_id) - EXTRACT(YEAR FROM cd.passenger_dateofbirth) BETWEEN 25 AND 34 THEN
'25-34'
        WHEN EXTRACT(YEAR FROM rf.date_id) - EXTRACT(YEAR FROM cd.passenger_dateofbirth) BETWEEN 35 AND 44 THEN
'35-44'
        ELSE '45+'
    END AS age_group,
    SUM(rf.revenue_amount) AS total_revenue,
    SUM(pf.profit_amount) AS total_profit
FROM RevenueFact rf
JOIN ProfitFact pf
    ON rf.flight_id = pf.flight_id
    AND rf.date_id = pf.date_id  -- Ensuring correct data mapping
JOIN customer_dim cd
    ON rf.passenger_id = cd.sk_passenger_id  -- Using `RevenueFact` to link passengers
GROUP BY
    CASE
        WHEN EXTRACT(YEAR FROM rf.date_id) - EXTRACT(YEAR FROM cd.passenger_dateofbirth) < 25 THEN '18-24'
        WHEN EXTRACT(YEAR FROM rf.date_id) - EXTRACT(YEAR FROM cd.passenger_dateofbirth) BETWEEN 25 AND 34 THEN
'25-34'
        WHEN EXTRACT(YEAR FROM rf.date_id) - EXTRACT(YEAR FROM cd.passenger_dateofbirth) BETWEEN 35 AND 44 THEN
'35-44'
        ELSE '45+'
    END
ORDER BY total_profit DESC;
```



| | AGE_GROUP | TOTAL_REVENUE | TOTAL_PROFIT |
|---|---|---|---|
| 1 | 35-44 | 2594584.15 | 1525812.43 |

**Based on these queries choose most appropriate indexes for better performance.**

| Table Name | Column Name | Index Type | Reason |
|---|---|---|---|
| customer_dim | sk_passenger_id | Primary - B-Tree | Used for quick lookups and joins. |
| customer_dim | passenger_status | Bitmap | Low-cardinality values (Gold, Platinum, etc.). Helps with filtering. |
| SegmentActivityFact | passenger_id | Foreign - B-Tree | Joins with customer_dim on sk_passenger_id. |
| SegmentActivityFact | flight_id | Foreign - B-Tree | Joins with flight_dim on flight_id. |
| SegmentActivityFact | class_services_id | Foreign - B-Tree | Joins with class_services_dim on class_of_services_id. |
| SegmentActivityFact | promotion_id | Foreign - B-Tree | Used to join with promotion_dim. |
| flight_dim | flight_id | Primary - B-Tree | Used for identifying flights and joins. |
| flight_dim | origin_airport_id, destination_airport_id | B-Tree | Searching for flights by airport needs efficient indexing. |
| ProfitFact | date_id | Foreign - B-Tree | Joins with date_dim for time-based analysis. |
| ProfitFact | flight_id | Foreign - B-Tree | Links profit data to specific flights. |
| ProfitFact | booking_channel_id | Foreign - B-Tree | Joins with booking_channel_dim to analyze revenue. |
| RevenueFact | passenger_id | Foreign - B-Tree | Links revenue data to passengers. |
| RevenueFact | promotion_id | Foreign - B-Tree | Used for evaluating promotions. |
| CustomerCareFact | employee_id | Foreign - B-Tree | Joins with employee_dim for performance tracking. |
| CustomerCareFact | interaction_type | Bitmap | Low-cardinality values (e.g., Complaint, Inquiry). |
| CustomerCareFact | customer_id | Foreign - B-Tree | Joins with customer_dim for performance tracking |

```
-- Indexes for SegmentActivityFact (Foreign Keys)
CREATE INDEX idx_segment_passenger ON SegmentActivityFact(passenger_id);
CREATE INDEX idx_segment_flight ON SegmentActivityFact(flight_id);
CREATE INDEX idx_segment_class ON SegmentActivityFact(class_services_id);
CREATE INDEX idx_segment_promotion ON SegmentActivityFact(promotion_id);

-- Indexes for ProfitFact (Foreign Keys)
CREATE INDEX idx_profit_flight ON ProfitFact(flight_id);
CREATE INDEX idx_profit_booking_channel ON ProfitFact(booking_channel_id);
CREATE INDEX idx_profit_promotion ON ProfitFact(promotion_id);
CREATE INDEX idx_profit_date ON ProfitFact(date_id);

-- Indexes for RevenueFact (Foreign Keys)
CREATE INDEX idx_revenue_passenger ON RevenueFact(passenger_id);
CREATE INDEX idx_revenue_promotion ON RevenueFact(promotion_id);

-- Indexes for CustomerCareFact (Foreign Keys)
CREATE INDEX idx_care_employee ON CustomerCareFact(employee_id);
CREATE INDEX idx_care_date ON CustomerCareFact(date_id);
create index idx_customer_care_feedback_id on customercarefact (feedback_id);
CREATE INDEX idx_customer_care_customer_id ON CustomerCareFact(customer_id);

-- Time dimension
CREATE INDEX idx_time_dim_hour ON time_dim(hour);
CREATE INDEX idx_time_dim_minutes ON time_dim(minute);

-- Flight dim
CREATE INDEX idx_flight_dim_origin_airport ON flight_dim(origin_airport_id);
CREATE INDEX idx_flight_dim_destination ON flight_dim(destination_airport_id);
CREATE INDEX idx_flight_dim_aircraft ON flight_dim(aircraft_id);
CREATE INDEX idx_flight_dim_origin_date ON flight_dim(origin_date);
CREATE INDEX idx_flight_dim_arrival_date ON flight_dim(arrival_date);

-- Composite Index for Passenger Data Queries
CREATE INDEX idx_passenger_status ON customer_dim(passenger_status);

-- Composite Index for Promotions and Revenue
CREATE INDEX idx_promotion_category ON promotion_dim(category);

-- Bitmap Index for Passenger Status (if low cardinality)
CREATE BITMAP INDEX bm_passenger_status ON customer_dim(passenger_status);

-- Function-Based Index for Date Queries
CREATE INDEX idx_date_month ON date_dim(TO_CHAR(date_id, 'YYYY-MM'));
```

| Index Type | Used On | Reason |
|---|---|---|
| **B-tree (default)** | Primary keys (PK) | Ensures uniqueness and fast lookups. |
| **B-tree** | Foreign keys (FK) | Speeds up join operations. |
| **Composite Index** | Queries filtering multiple columns | Optimizes search performance. |
| **Bitmap Index** | Low-cardinality columns (e.g., passenger_status) | Improves filtering efficiency. |
| **Function-Based Index** | TO_CHAR(date_id, 'YYYY-MM') | Optimizes date-based searches. |