Information Systems Assignment

Ayano Yamamoto

30th December 2021

Table of Contents

A. Conceptual Design

a. Full list of requirements and assumptions

Requirements

An airline has requested a database to be designed. Their initial requirements are:

- The airline operates direct flight routes between its hub (Dublin Airport) and other European airports.

- Each airport is uniquely identified by an IATA (the International Air Transport Association) airport code.

- Each route is uniquely identified by a flight number. For example, flights from Dublin Airport to Heathrow Airport are EI 158. Flights from Heathrow Airport to Dublin airport are EI 159.

- The airline creates an operation schedule with each flight number operating maximum of once per day. The specific combination of a date and a flight number is uniquely identified by a schedule ID.

- Each schedule ID has a departure time, arrival time, and a base price of the flight.

- One model of airplane is used for all flights. Airplane seats are uniquely identified by seat numbers which indicate the class of the seats.

- Each class are allocated a price multiplier which is used to multiply the base prices by to determine the seat price. For example, if the base price of a specific flight is €100.00, and the business class price multiplier is 2, the price of a business class seat on this flight is €200.00.

- Availability of each seat per flight is recorded in either “0” (reserved) or “1” (available).

Assumptions

- Only one person uses the database.
- All flights have the same seat arrangements, including the classes and their multipliers.
- Departure times and arrival times are recorded in the local times of corresponding airports.
- There are no flights operated over midnight. All flights depart and arrive within the same day.
- Each route is scheduled to fly at least once in the database.
- Not all airports are located in the same time zones.
- All prices in the database are in Euro. There is no currency conversion.
- There is no discount offered for children of any age.
- Reservation details such as passenger and payment information are out of scope for this database.

b. Rough ER diagrams

Entities identified from the initial requirements are Airports, Routes, Schedules, Seats, and Classes. Airlines and Airplane Models could also be possible entities, but they are not included since there will only be one airline and one model of airplane in this database.

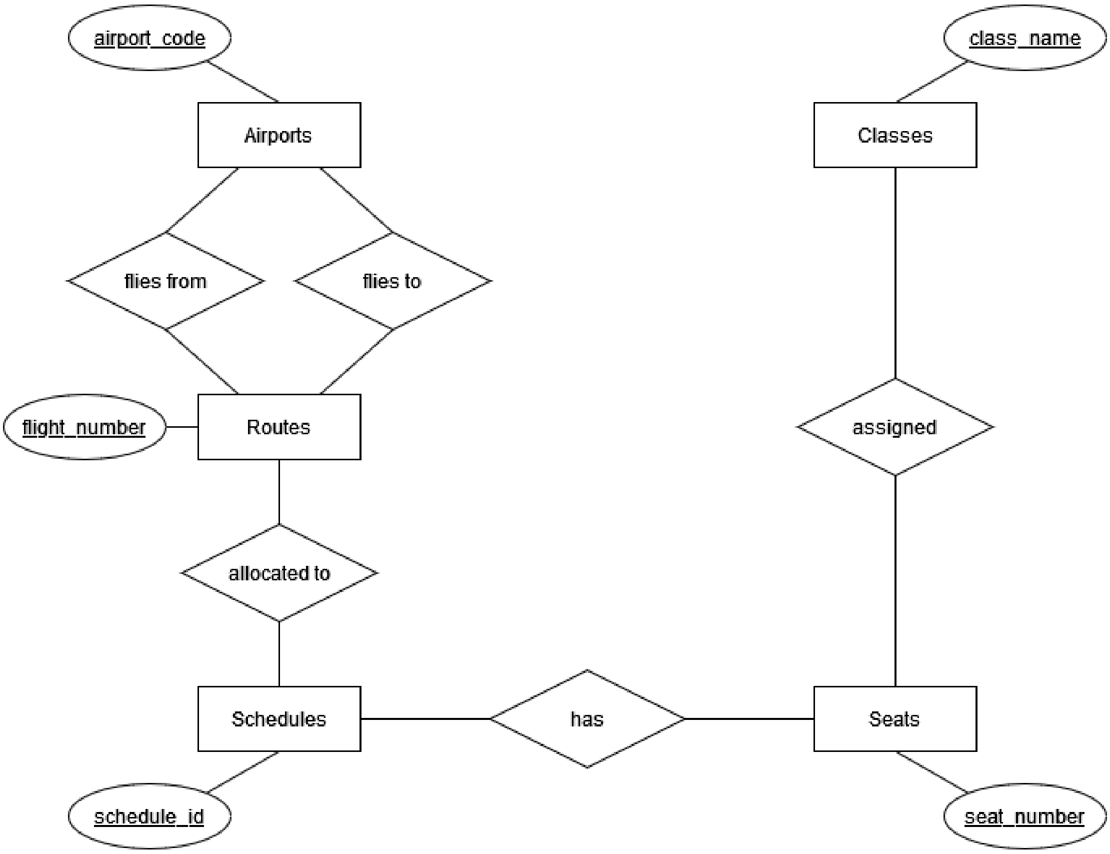


Figure 1: First Rough ER Diagram with Relationships

Multiple relationships are identified between airports and routes, where each route has one airport it flies from, and another airport it flies to. Each relationship will be handled separately in logical design.

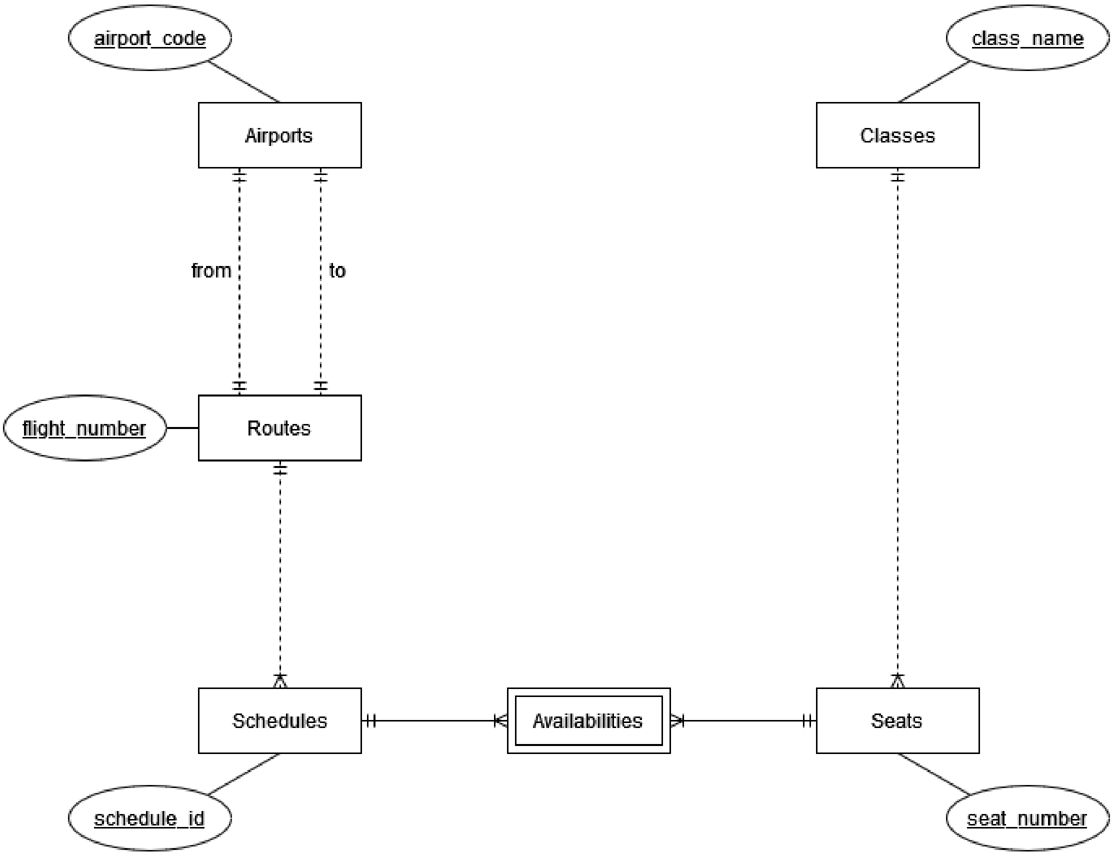


Figure 2: Second Rough ER Diagram with a Bridge Entity

- Airports-to-Routes are two One-to-One relationships.

- One route can historically be scheduled several times. Routes-to-Schedules is a One-to-Many relationship.

- One class can be assigned to several seats. Classes-to-Seats is a One-to-Many relationship.

- One seat number can be booked in many scheduled flights, and one scheduled flight can have many seats. The Many-to-Many relationship between Schedules-to-Seats cannot be translated directly into a relational database design. A bridge entity Availabilities is introduced to build a bridge between the two entities.

c. Final key-based ER diagram

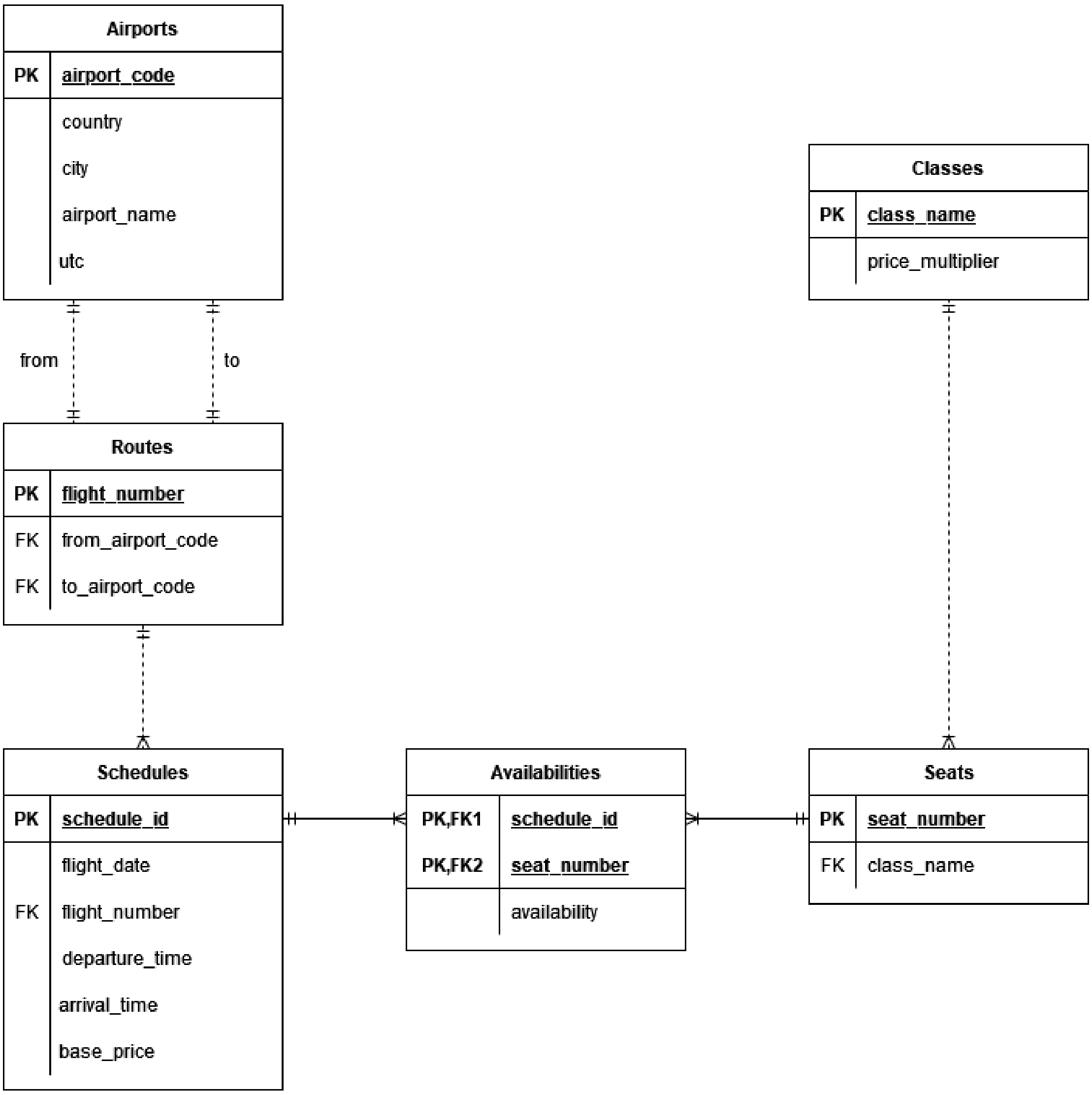


Figure 3: Fully Attributed ER Diagram

Attributes identified from the initial requirements are:

Entities	Attributes
Airports	airport_code, city
Routes	flight_number, from_airport_code, to_airport_code
Schedules	schedule_id, flight_date, flight_number, departure_time, arrival_time, base_price
Availabilities	schedule_id, seat_number, availability
Seats	Seat_number, class_name
Classes	class_name, price_multiplier

Additional attributes country and airport_name have been included in the Airports entity, together with UTC (Coordinated Universal Time) of the airport location in order to calculate durations of the flights.

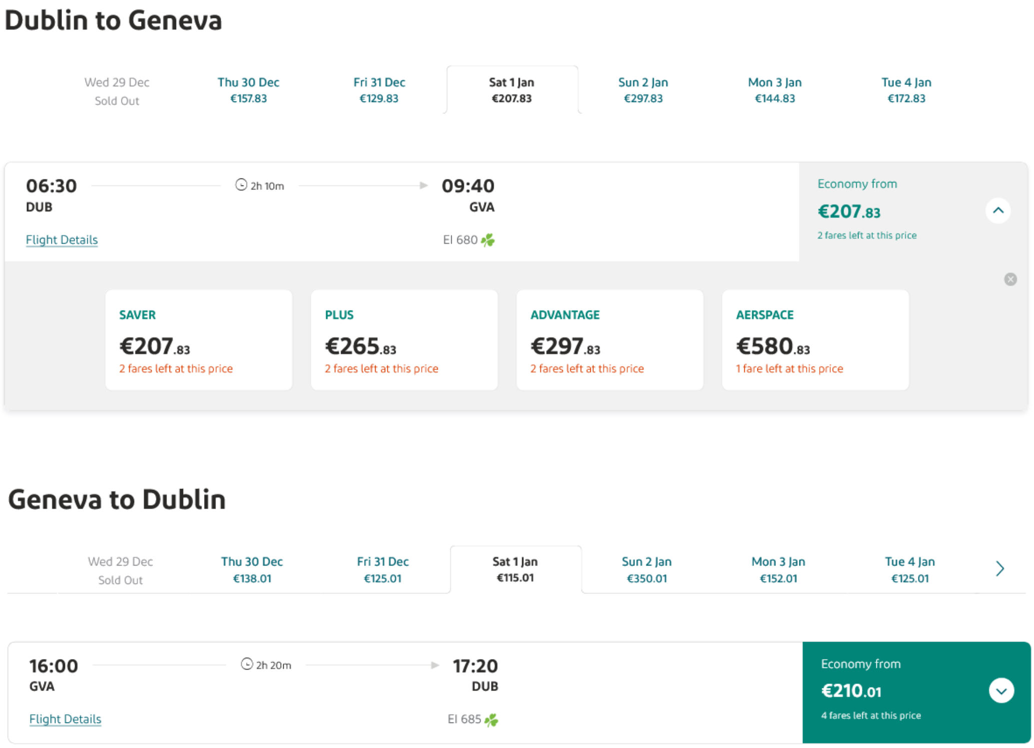
B. *Logical Design*

a. Final design in 3NF

Logical Schema:

Schedules-3 (**schedule_id**, flight_date, departure_time, arrival_time, base_price, flight_number)
Routes-3 (**flight_number**, from_airport_code, to_airport_code)
Airports-3 (**airport_code**, city, country, airport_name, utc)
Availabilities-3 (**schedule_id**, **seat_number**, availability)
Seats-3 (**seat_number**, class_name)
Classes-3 (**class_name**, price_multiplier)

b. Screenshot



Dublin to Geneva

Wed 29 Dec	Thu 30 Dec	Fri 31 Dec	Sat 1 Jan	Sun 2 Jan	Mon 3 Jan	Tue 4 Jan
Sold Out	€157.83	€129.83	€207.83	€297.83	€144.83	€172.83

06:30 DUB — 2h 10m — 09:40 GVA
Flight Details
EI 680

Economy from
€207.83
2 fares left at this price

SAVER	PLUS	ADVANTAGE	AERSPACE
€207.83	€265.83	€297.83	€580.83
2 fares left at this price	2 fares left at this price	2 fares left at this price	1 fare left at this price

Geneva to Dublin

Wed 29 Dec	Thu 30 Dec	Fri 31 Dec	Sat 1 Jan	Sun 2 Jan	Mon 3 Jan	Tue 4 Jan
Sold Out	€138.01	€125.01	€115.01	€350.01	€152.01	€125.01

16:00 GVA — 2h 20m — 17:20 DUB
Flight Details
EI 685

Economy from
€210.01
4 fares left at this price

Figure 4: Search Results Screen for Available Flights

c. Normalisation steps

UNF

Schedules (schedule_id, flight_date, weekday, departure_time, arrival_time, duration, base_price, flight_number, from_airport_code, to_airport_code, from_city, to_city, from_country, to_country, from_airport_name, to_airport_name, from_airport_utc, to_airport_utc, (seat_number, class_name, price_multiplier, availability)*)

Attributes are gathered with seat availability identified as repeating attributes.

Weekday is identified as a calculated attribute since it can be calculated from flight_date.

Duration is also identified as a calculated attribute, since it can be calculated from departure_time, arrival_time, from_airport_utc, and to_airport_utc.

1NF

Schedules-1 (**schedule_id**, flight_date, departure_time, arrival_time, base_price, flight_number, from_airport_code, to_airport_code, from_city, to_city, from_country, to_country, from_airport_name, to_airport_name, from_airport_utc, to_airport_utc)

Availabilities-1 (**schedule_id**, **seat_number**, class_name, price_multiplier, availability)

Calculated attributes are removed.

Repeating attributes are removed to Availabilties-1 entity with both schedule_id and seat_number as the Primary Keys.

2NF

Schedules-2 (**schedule_id**, flight_date, departure_time, arrival_time, base_price, flight_number, from_airport_code, to_airport_code, from_city, to_city, from_country, to_country, from_airport_name, to_airport_name, from_airport_utc, to_airport_utc)

Availabilities-2 (**schedule_id**, **seat_number**, availability)

Seats-2 (**seat_number**, class_name, price_multiplier)

Class_name and price_multipler have a partial functional depencency on seat_number. They have been removed to the Seats-2 entity with seat_number as the Primary Key.

3NF

Schedules-2 (**schedule_id**, flight_date, departure_time, arrival_time, base_price, flight_number, from_airport_code, to_airport_code, from_city, to_city, from_country, to_country, from_airport_name, to_airport_name, from_airport_utc, to_airport_utc)

Availabilities-3 (**schedule_id**, **seat_number**, availability)

Seats-3 (**seat_number**, class_name)

Classes-3 (**class_name**, price_multiplier)

Price_multiplier has a transitive dependency on class_name. It has been removed to Classes-3 entity with class_name as the Primary Key.

Schedules-3 (**schedule_id**, flight_date, departure_time, arrival_time, base_price, flight_number)

Routes-3 (**flight_number**, from_airport_code, to_airport_code, from_city, to_city, from_country, to_country, from_airport_name, to_airport_name, from_airport_utc, to_airport_utc)

Availabilities-3 (**schedule_id**, **seat_number**, availability)

Seats-3 (**seat_number**, class_name)

Classes-3 (**class_name**, price_multiplier)

Departure and arrival airport information have a transitive dependency on flight_number. They have been removed to Routes-3 entity with flight_number as the Primary Key.

Schedules-3 (**schedule_id**, flight_date, departure_time, arrival_time, base_price, flight_number)

Routes-3 (**flight_number**, from_airport_code, to_airport_code)

Airports-3 (**airport_code**, city, country, airport_name, utc)

Availabilities-3 (**schedule_id**, **seat_number**, availability)

Seats-3 (**seat_number**, class_name)

Classes-3 (**class_name**, price_multiplier)

Airport information have further transitive dependency on airport_code. They have been removed to Airports-3 entity with airport_code as the Primary Key.

C. Physical Design

a. MySQL Implementation

Tables

The database has six tables below.

- Airports

```
CREATE TABLE airports (
            airport_code VARCHAR(3) NOT NULL,
            counrty VARCHAR(20) NOT NULL,
            city VARCHAR(20) NOT NULL,
            airport_name VARCHAR(50) NOT NULL,
            utc TIME NOT NULL,
            PRIMARY KEY (airport_code),
            UNIQUE (airport_code)
);
```

The Airports table stores UTC in time format. For example, airport_code BER (Berlin Brandenberg Airport) is in UTC +1, hence it has a utc value of 01:00:00.

- Routes

```
CREATE TABLE routes (
            flight_number VARCHAR(6) NOT NULL,
            from_airport_code VARCHAR(3) NOT NULL,
            to_airport_code VARCHAR(3) NOT NULL,
            PRIMARY KEY (flight_number),
            UNIQUE (flight_number),
            FOREIGN KEY (from_airport_code) REFERENCES airports (airport_code),
            FOREIGN KEY (to_airport_code) REFERENCES airports (airport_code)
);
```

The Routes table has two foreign keys from_airport_code and to_airport_code. They both reference airport_code from the Airports table.

- Schedules

```
CREATE TABLE schedules (
        schedule_id SMALLINT NOT NULL AUTO_INCREMENT,
        flight_date DATE NOT NULL,
        flight_number VARCHAR(6) NOT NULL,
        departure_time TIME NOT NULL,
        arrival_time TIME NOT NULL,
        base_price FLOAT NOT NULL,
        PRIMARY KEY (schedule_id),
        UNIQUE (schedule_id),
        FOREIGN KEY (flight_number) REFERENCES routes(flight_number)
);
```

The Schedules table has auto incremented value for schedule_id. Departure_time and arrival_time are both in time format, so they can be used to calculate flight durations using UTC.

- Classes

```
CREATE TABLE classes (
        class_name VARCHAR(15) NOT NULL,
        price_multiplier FLOAT NOT NULL,
        PRIMARY KEY (class_name),
        UNIQUE (class_name)
);
```

The Classes table stores price_multiplier in float format, so it can be used to calculate seat prices using base_price from the Schedules table.

- Seats

```
CREATE TABLE seats (
        seat_number VARCHAR(3) NOT NULL,
        class_name VARCHAR(15) NOT NULL,
        PRIMARY KEY (seat_number),
        UNIQUE (seat_number),
        FOREIGN KEY (class_name) REFERENCES classes (class_name)
);
```

Seats table assigns class names to seat arrangements in the airplane.

- Availabilities

```
CREATE TABLE availabilities (
	schedule_id SMALLINT NOT NULL,
	seat_number VARCHAR(3) NOT NULL,
	availability TINYINT(1) NOT NULL,
	PRIMARY KEY (schedule_id, seat_number),
	FOREIGN KEY (schedule_id) REFERENCES schedules(schedule_id),
	FOREIGN KEY (seat_number) REFERENCES seats(seat_number)
);
```

The Availabilities table stores seat availabilities in tinyint values of 0 or 1 to simulate Boolean type data. It has two primary keys and two foreign keys of schedule_id and seat_number.

Views

- flight_details

```
CREATE VIEW flight_details AS
        SELECT sc.flight_date AS flight_date,
                DAYNAME(sc.flight_date) AS dayname,
                sc.flight_number AS flight_number,
                r.from_airport_code AS from_airport_code,
                from_airports.city AS from_city,
                r.to_airport_code AS to_airport_code,
                to_airports.city AS to_city,
                sc.departure_time AS departure_time,
                sc.arrival_time AS arrival_time,
                CAST(TIMEDIFF((sc.arrival_time - to_airports.utc),(sc.departure_time - from_airports.utc)) AS TIME) AS duration,
                CAST((sc.base_price * c.price_multiplier) AS DECIMAL (7,2)) AS price,
                se.class_name AS class,
                COUNT(av.availability) AS available_seats
        FROM schedules sc
                JOIN routes r USING (flight_number)
                JOIN availabilities av USING (schedule_id)
                JOIN seats se USING (seat_number)
                JOIN classes c USING (class_name)
                LEFT JOIN airports from_airports ON r.from_airport_code = from_airports.airport_code
                LEFT JOIN airports to_airports ON r.to_airport_code = to_airports.airport_code
        WHERE av.availability = TRUE
        GROUP BY sc.flight_date,
                sc.flight_number,
                se.class_name;
```

The flight_details view joins all six tables in the database and displays the information included in the Screenshot above. Since we're looking for information on available seats, it is filtered for rows where availability value is TRUE (1).

The calculated fields are:

- **dayname**: calculated from flight_date
- **duration**: calculated as the difference between departure_time and arrival_time, adjusted for time difference using UTCs of from_airports and to_airports
- **price**: calculated as the multiplication of base_price and price_multiplier
- **available_seats**: count of availability

- flight_revenues

```
CREATE VIEW flight_revenues AS
        SELECT sc.schedule_id,
                sc.flight_date,
                sc.flight_number,
                CAST(SUM((sc.base_price * c.price_multiplier)) AS DECIMAL(7,2))
                AS revenue_per_flight
        FROM availabilities av
                JOIN schedules sc USING (schedule_id)
                JOIN seats se USING (seat_number)
                JOIN classes c USING (class_name)
        WHERE av.availability = FALSE
        GROUP BY sc.flight_date,
                sc.flight_number;
```

The flight_revenues view joins four tables to display revenues of each scheduled flight. Since revenue is calculated from reserved seats, it is filtered for rows where availability value is FALSE (0).

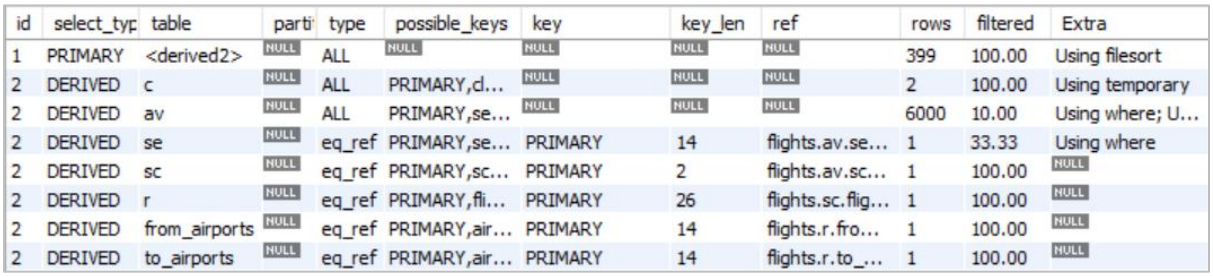
The calculated field is:

- **revenue_per_flight**: sum of seat prices, which are calculated by base_price multiplied by price_multiplier

b. Indexing

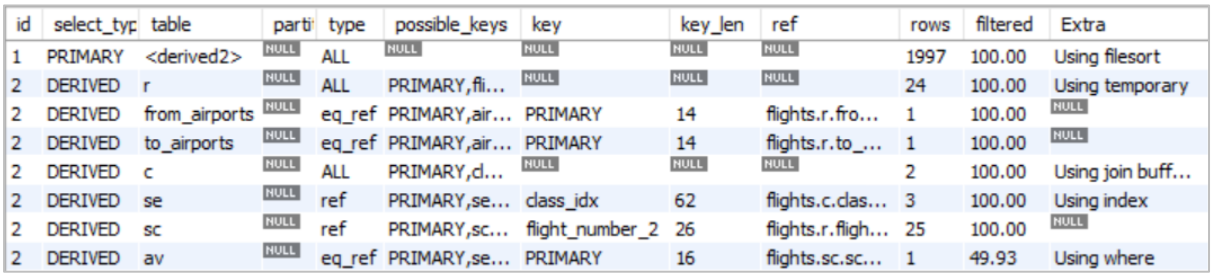
Index is added to availability column from availabilities table.

In an example of Query 2, the index decreases the total number of rows examined from 6,406 to 2,054, reducing it by 67.9%.



id	select_typ	table	parti	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	PRIMARY	<derived2>	NULL	ALL	NULL	NULL	NULL	NULL	399	100.00	Using filesort
2	DERIVED	c	NULL	ALL	PRIMARY,cl…	NULL	NULL	NULL	2	100.00	Using temporary
2	DERIVED	av	NULL	ALL	PRIMARY,se…	NULL	NULL	NULL	6000	10.00	Using where; U…
2	DERIVED	se	NULL	eq_ref	PRIMARY,se…	PRIMARY	14	flights.av.se…	1	33.33	Using where
2	DERIVED	sc	NULL	eq_ref	PRIMARY,sc…	PRIMARY	2	flights.av.sc…	1	100.00	NULL
2	DERIVED	r	NULL	eq_ref	PRIMARY,fli…	PRIMARY	26	flights.sc.flig…	1	100.00	NULL
2	DERIVED	from_airports	NULL	eq_ref	PRIMARY,air…	PRIMARY	14	flights.r.fro…	1	100.00	NULL
2	DERIVED	to_airports	NULL	eq_ref	PRIMARY,air…	PRIMARY	14	flights.r.to_…	1	100.00	NULL

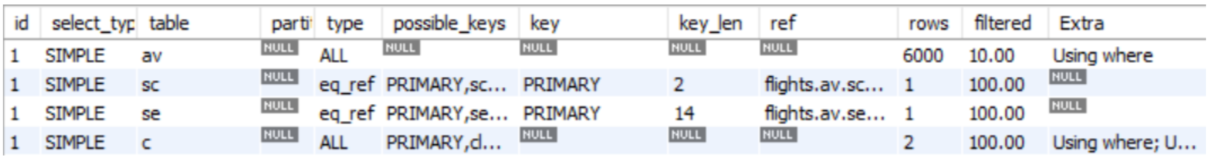
Figure 5: Explain Results of Query 2 without Index



id	select_typ	table	parti	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	PRIMARY	<derived2>	NULL	ALL	NULL	NULL	NULL	NULL	1997	100.00	Using filesort
2	DERIVED	r	NULL	ALL	PRIMARY,fli…	NULL	NULL	NULL	24	100.00	Using temporary
2	DERIVED	from_airports	NULL	eq_ref	PRIMARY,air…	PRIMARY	14	flights.r.fro…	1	100.00	NULL
2	DERIVED	to_airports	NULL	eq_ref	PRIMARY,air…	PRIMARY	14	flights.r.to_…	1	100.00	NULL
2	DERIVED	c	NULL	ALL	PRIMARY,cl…	NULL	NULL	NULL	2	100.00	Using join buff…
2	DERIVED	se	NULL	ref	PRIMARY,se…	class_idx	62	flights.c.clas…	3	100.00	Using index
2	DERIVED	sc	NULL	ref	PRIMARY,sc…	flight_number_2	26	flights.r.fligh…	25	100.00	NULL
2	DERIVED	av	NULL	eq_ref	PRIMARY,se…	PRIMARY	16	flights.sc.sc…	1	49.93	Using where

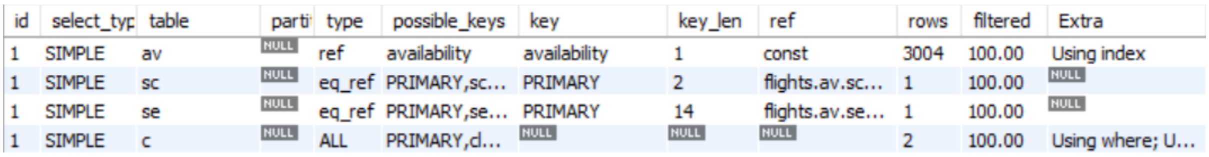
Figure 6: Explain Results of Query 2 with Index

In another example of Query 5, the index decreases the total number of rows examined from 6,004 to 3,008, reducing it by 49.9%.



id	select_typ	table	parti	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	av	NULL	ALL	NULL	NULL	NULL	NULL	6000	10.00	Using where
1	SIMPLE	sc	NULL	eq_ref	PRIMARY,sc…	PRIMARY	2	flights.av.sc…	1	100.00	NULL
1	SIMPLE	se	NULL	eq_ref	PRIMARY,se…	PRIMARY	14	flights.av.se…	1	100.00	NULL
1	SIMPLE	c	NULL	ALL	PRIMARY,cl…	NULL	NULL	NULL	2	100.00	Using where; U…

Figure 7: Explain Results of Query 5 without Index



id	select_typ	table	parti	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	av	NULL	ref	availability	availability	1	const	3004	100.00	Using index
1	SIMPLE	sc	NULL	eq_ref	PRIMARY,sc…	PRIMARY	2	flights.av.sc…	1	100.00	NULL
1	SIMPLE	se	NULL	eq_ref	PRIMARY,se…	PRIMARY	14	flights.av.se…	1	100.00	NULL
1	SIMPLE	c	NULL	ALL	PRIMARY,cl…	NULL	NULL	NULL	2	100.00	Using where; U…

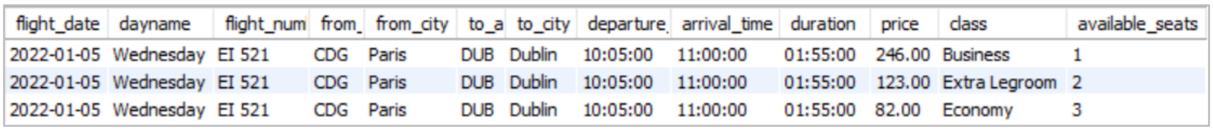
Figure 8: Explain Results of Query 5 with Index

c. Sample queries

Query 1: One-Way Flight, One Passenger, Paris to Dublin

Query 1 simulates a search for one-way flight for one passenger for a particular date. It uses Regular Expression to look for city names that start with ‘pa’ and ‘du.’

```
SELECT fd.*
FROM flight_details fd
WHERE flight_date = '2022-01-05'
            AND REGEXP_LIKE (from_city, '^pa', 'i')
            AND REGEXP_LIKE (to_city, '^du', 'i');
```



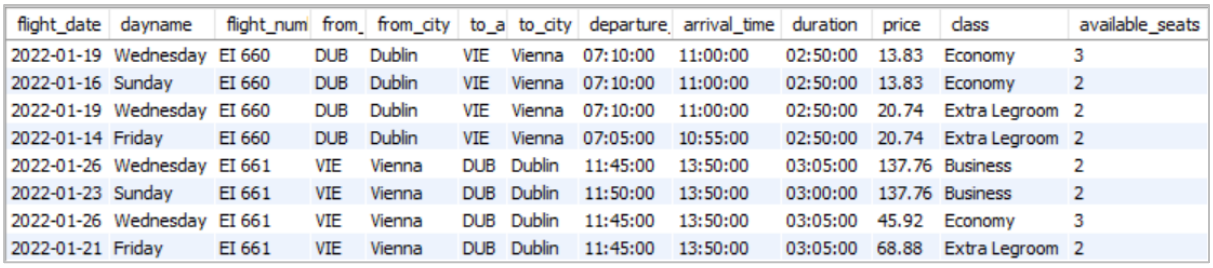
flight_date	dayname	flight_num	from_	from_city	to_a	to_city	departure_	arrival_time	duration	price	class	available_seats
2022-01-05	Wednesday	EI 521	CDG	Paris	DUB	Dublin	10:05:00	11:00:00	01:55:00	246.00	Business	1
2022-01-05	Wednesday	EI 521	CDG	Paris	DUB	Dublin	10:05:00	11:00:00	01:55:00	123.00	Extra Legroom	2
2022-01-05	Wednesday	EI 521	CDG	Paris	DUB	Dublin	10:05:00	11:00:00	01:55:00	82.00	Economy	3

Figure 9: Results of Query 1

Query 2: Return Flight, Two Passengers, Dublin to Vienna

Query 2 simulates a search for return flights for 2 passengers with flexible dates. It shows flights with more than 2 available seats in the same class, with date range of 7 days for both flights.

```
SELECT fd.*
FROM flight_details fd
WHERE (flight_date BETWEEN '2022-01-13' AND '2022-01-19')
            AND from_airport_code = 'DUB'
            AND to_airport_code = 'VIE'
            AND available_seats >= 2
            OR (flight_date BETWEEN '2022-01-20' AND '2022-01-26')
            AND from_airport_code = 'VIE'
            AND to_airport_code = 'DUB'
            AND available_seats >= 2
ORDER BY flight_number ASC;
```



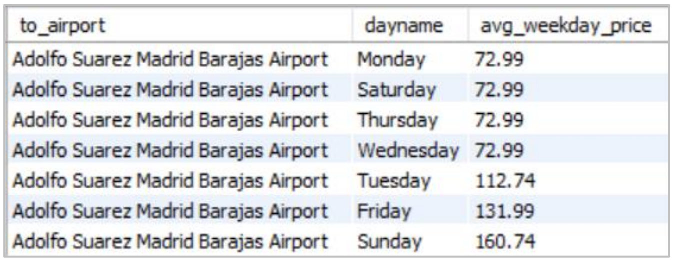
flight_date	dayname	flight_num	from_	from_city	to_a	to_city	departure_	arrival_time	duration	price	class	available_seats
2022-01-19	Wednesday	EI 660	DUB	Dublin	VIE	Vienna	07:10:00	11:00:00	02:50:00	13.83	Economy	3
2022-01-16	Sunday	EI 660	DUB	Dublin	VIE	Vienna	07:10:00	11:00:00	02:50:00	13.83	Economy	2
2022-01-19	Wednesday	EI 660	DUB	Dublin	VIE	Vienna	07:10:00	11:00:00	02:50:00	20.74	Extra Legroom	2
2022-01-14	Friday	EI 660	DUB	Dublin	VIE	Vienna	07:05:00	10:55:00	02:50:00	20.74	Extra Legroom	2
2022-01-26	Wednesday	EI 661	VIE	Vienna	DUB	Dublin	11:45:00	13:50:00	03:05:00	137.76	Business	2
2022-01-23	Sunday	EI 661	VIE	Vienna	DUB	Dublin	11:50:00	13:50:00	03:00:00	137.76	Business	2
2022-01-26	Wednesday	EI 661	VIE	Vienna	DUB	Dublin	11:45:00	13:50:00	03:05:00	45.92	Economy	3
2022-01-21	Friday	EI 661	VIE	Vienna	DUB	Dublin	11:45:00	13:50:00	03:05:00	68.88	Extra Legroom	2

Figure 10: Results of Query 2

Query 3: From Cheapest to Most Expensive Days to Fly Business Class to Madrid

Query 3 searches for available business class seats on flights between IATA airport code ‘DUB’ and airport names including 'suarez' or 'suerez'. It then calculates the average distinct prices for every day of the week, and orders them from the cheapest to the most expensive.

```
SELECT to_airports.airport_name AS to_airport,
        fd.dayname,
        CAST(AVG(DISTINCT(fd.price)) AS DECIMAL(7,2)) AS avg_weekday_price
FROM airports to_airports
        RIGHT JOIN flight_details fd ON fd.to_airport_code = to_airports.airport_code
WHERE fd.class = 'Business'
        AND fd.from_airport_code = 'DUB'
        AND REGEXP_LIKE (to_airports.airport_name, 'su(a|e)rez', ‘i’)
GROUP BY fd.dayname,
        to_airports.airport_name
ORDER BY avg_weekday_price ASC;
```



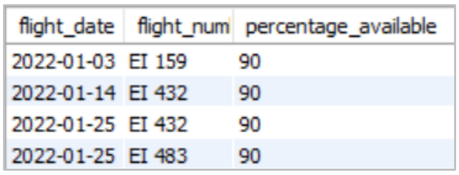
to_airport	dayname	avg_weekday_price
Adolfo Suarez Madrid Barajas Airport	Monday	72.99
Adolfo Suarez Madrid Barajas Airport	Saturday	72.99
Adolfo Suarez Madrid Barajas Airport	Thursday	72.99
Adolfo Suarez Madrid Barajas Airport	Wednesday	72.99
Adolfo Suarez Madrid Barajas Airport	Tuesday	112.74
Adolfo Suarez Madrid Barajas Airport	Friday	131.99
Adolfo Suarez Madrid Barajas Airport	Sunday	160.74

Figure 10: Results of Query 3

Query 4: Flights with 90% Seats Availability

Query 4 divides the sum of availability by the count of availability to calculate the percentage of available seats for every flight. It then shows flights with availabilities more than or equal to 90%.

```
SELECT sc.flight_date,
        sc.flight_number,
        FORMAT((SUM(av.availability) / COUNT(av.availability))*100, 0) AS
        percentage_available
FROM schedules sc
        INNER JOIN availabilities av USING (schedule_id)
GROUP BY sc.flight_date,
        sc.flight_number
HAVING percentage_available >= 90;
```



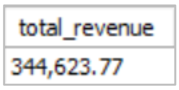
flight_date	flight_num	percentage_available
2022-01-03	EI 159	90
2022-01-14	EI 432	90
2022-01-25	EI 432	90
2022-01-25	EI 483	90

Figure 11: Results of Query 4

Query 5: Total Revenue

Query 5 calculates the total revenue of all flights in the database by adding all the seat prices of reserved seats.

```
SELECT FORMAT(SUM(sc.base_price * c.price_multiplier), 2) AS total_revenue
FROM availabilities av
        LEFT JOIN schedules sc ON av.schedule_id = sc.schedule_id
        LEFT JOIN seats se ON av.seat_number = se.seat_number
        LEFT JOIN classes c ON se.class_name = c.class_name
WHERE av.availability = FALSE;
```



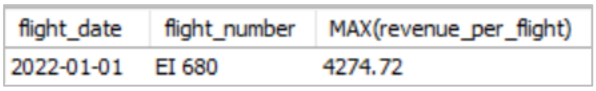
total_revenue
344,623.77

Figure 12: Result of Query 5

Query 6: Single Flight with Highest Revenue

Query 6 shows the flight date and flight number of the single flight with the highest revenue in the database.

```
SELECT flight_date,
        flight_number,
        MAX(revenue_per_flight)
FROM flight_revenues;
```



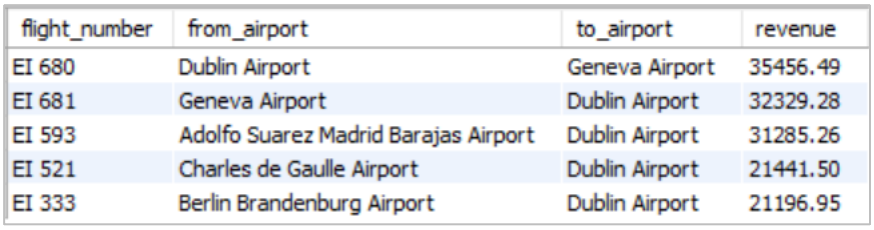
flight_date	flight_number	MAX(revenue_per_flight)
2022-01-01	EI 680	4274.72

Figure 11: Results of Query 6

Query 7: Top 5 Highest Revenue Routes with Airport Names

Query 7 joins flight_revenues view with the Airports table to display airport names for the top 5 highest revenue routes and their total revenues.

```
SELECT fr.flight_number,
        airport_names.from_airport,
        airport_names.to_airport,
        SUM(fr.revenue_per_flight) AS revenue
FROM flight_revenues fr
        LEFT JOIN(
        SELECT r.flight_number,
                from_airports.airport_name AS from_airport,
                to_airports.airport_name AS to_airport
        FROM routes r
                LEFT JOIN airports from_airports ON r.from_airport_code =
                from_airports.airport_code
                LEFT JOIN airports to_airports ON r.to_airport_code =
                to_airports.airport_code
        ) AS airport_names ON fr.flight_number = airport_names.flight_number
GROUP BY flight_number
ORDER BY revenue DESC
LIMIT 5;
```



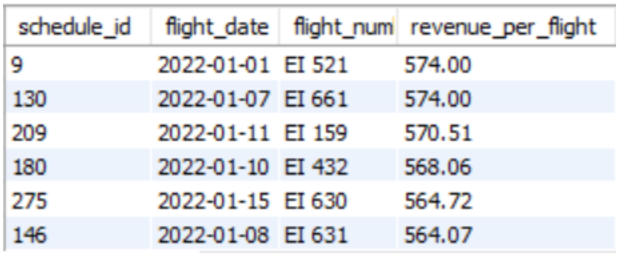
flight_number	from_airport	to_airport	revenue
EI 680	Dublin Airport	Geneva Airport	35456.49
EI 681	Geneva Airport	Dublin Airport	32329.28
EI 593	Adolfo Suarez Madrid Barajas Airport	Dublin Airport	31285.26
EI 521	Charles de Gaulle Airport	Dublin Airport	21441.50
EI 333	Berlin Brandenburg Airport	Dublin Airport	21196.95

Figure 12: Results of Query 7

Query 8: Routes with Below Average Revenue

Query 8 calculates the average total revenue of all flights in the database, and displays flights with below average revenues in descending order.

```
SELECT *
FROM flight_revenues
WHERE revenue_per_flight < (SELECT AVG(revenue_per_flight) FROM
flight_revenues)
ORDER BY revenue_per_flight DESC;
```



schedule_id	flight_date	flight_num	revenue_per_flight
9	2022-01-01	EI 521	574.00
130	2022-01-07	EI 661	574.00
209	2022-01-11	EI 159	570.51
180	2022-01-10	EI 432	568.06
275	2022-01-15	EI 630	564.72
146	2022-01-08	EI 631	564.07

Figure 13: Part of the Results of Query 8 (total 377 rows)