

Project Final Report

ระบบจองตั๋วเครื่องบิน (FlyWithSigma)

เสนอ

รศ.ดร.ดวงดาว วิชาตากุล

รายชื่อผู้จัดทำ

คณาพงษ์ คำชุม 6631302421

ญาดา ะสีนนท์ 6631311021

ฉัตริน ยุ่นฉลาด 6631304721

ชยพล กุลตานนท์ 6631306021

ชยุต อาขามงคล 6631307621

ชนัดดา คนชม 6631305321

สุวิจักขณ์ จินตานพันธ์ 6631356321

บุญญพัฒน์ กิจวรชัย 6631334521

นันทพร พัวพันบุญ 6631330021

โครงการนี้เป็นส่วนหนึ่งของรายวิชา 2110322 ระบบฐานข้อมูล

ภาคการศึกษาปลาย ปีการศึกษา 2567

ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์

จุฬาลงกรณ์มหาวิทยาลัย

สารบัญ

ความเป็นมาของโครงการ	3
วัตถุประสงค์ของโครงการ	4
ฟังก์ชันของระบบ (System Functionalities)	5
ER Diagram	8
Schema Diagram (Referential Integrity)	9
Normalization (3NF)	10
Data Dictionary	11
DDL IN APPLICATION: INSERT, DELETE, UPDATE	20
Indexing	22
Stored Procedures	23
Stored Functions	27
Triggers	31
Execution Path	33
SQL Complex Queries	35
Document-based Design Schema	38
MongoDB	41

ความเป็นมาของโครงการ

ปัจจุบันการเดินทางโดยเครื่องบินเป็นหนึ่งในวิธีการเดินทางที่ได้รับความนิยมสูงสุด เนื่องจากสามารถเดินทางได้รวดเร็วและสะดวกสบาย อย่างไรก็ตาม กระบวนการจองตั๋วเครื่องบินผ่านช่องทางต่าง ๆ อาจมีความซับซ้อน เช่น การเปรียบเทียบราคาเที่ยวบิน การเลือกเที่ยวบินที่เหมาะสม หรือปัญหาด้านการชำระเงิน เพื่อแก้ไขปัญหาเหล่านี้ โครงการ **FlyWithSigma** จึงถูกพัฒนาขึ้นมาเพื่อเป็นระบบจองตั๋วเครื่องบินออนไลน์ที่ช่วยให้ผู้ใช้งานสามารถค้นหาเที่ยวบินที่เหมาะสมได้ง่ายและสะดวกขึ้น ผ่านการแสดงข้อมูลเที่ยวบินที่ครบถ้วน ทั้งราคาค่าโดยสาร ข้อมูลสนามบิน รายละเอียดของเที่ยวบิน รวมไปถึงตัวเลือกการชำระเงินที่หลากหลาย

นอกจากนี้ **FlyWithSigma** ยังรองรับการให้บริการหลังการจอง เช่น การเรียกดูตั๋วอิเล็กทรอนิกส์ (e-ticket) การตรวจสอบประวัติการจอง และการส่งเรื่องร้องเรียนเพื่อแจ้งปัญหาหรือข้อเสนอแนะต่อผู้ให้บริการสายการบิน ระบบนี้ได้รับการออกแบบมาเพื่อให้ตอบโจทย์ทั้งผู้ใช้งานทั่วไปและแอดมินที่ดูแลระบบ เพื่อให้การจองตั๋วเป็นไปอย่างมีประสิทธิภาพและโปร่งใส

วัตถุประสงค์ของโครงการ

1. พัฒนาระบบจองตั๋วเครื่องบินออนไลน์ ให้ผู้ใช้งานสามารถค้นหาเที่ยวบิน เปรียบเทียบราคา และจองตั๋วได้อย่างสะดวกและรวดเร็ว
2. เพิ่มความสะดวกในการชำระเงิน โดยรองรับทั้ง eBanking / บัตรเครดิต / บัตรเดบิต / PayPal เพื่อให้ผู้ใช้งานสามารถเลือกวิธีที่เหมาะสมกับตนเอง
3. จัดเก็บข้อมูลผู้ใช้และประวัติการจอง เพื่อให้สามารถเข้าถึงข้อมูลตัวที่จองไว้ได้ทุกที่ทุกเวลารวมถึงเรียกดู e-ticket ได้ง่าย
4. สนับสนุนการสื่อสารระหว่างผู้ใช้งานและแอดมิน ผ่านระบบร้องเรียน (report) เพื่อให้สามารถแจ้งปัญหาหรือข้อเสนอแนะเกี่ยวกับการจองหรือเที่ยวบินได้
5. ช่วยให้แอดมินสามารถบริหารจัดการข้อมูลเที่ยวบินและผู้โดยสารได้อย่างมีประสิทธิภาพ เช่น การส่งรายงานรายชื่อผู้โดยสารไปยังสายการบิน หรือจัดการเรื่องร้องเรียนที่ได้รับ
6. เพิ่มความน่าเชื่อถือและความปลอดภัยในการใช้งานระบบ โดยกำหนดให้ผู้ใช้ต้องลงทะเบียนและยืนยันตัวตนก่อนทำการจองเที่ยวบิน

ฟังก์ชันของระบบ (System Functionalities)

1. ส่วนการจัดการบัญชีผู้ใช้งาน

- ระบบต้องรองรับการสมัครบัญชีผู้ใช้
- ระบบต้องรองรับการกรอกข้อมูลพื้นฐานของผู้ใช้ เมื่อทำการสมัครบัญชี ได้แก่ ชื่อจริง นามสกุล ใต้อีเมล รหัสผ่าน เลขประจำตัวประชาชน เลขพาสปอร์ต อีเมล และเบอร์โทรศัพท์
- ระบบต้องสามารถเก็บข้อมูลของผู้ใช้ได้
- ระบบต้องรองรับการเปลี่ยนข้อมูลของผู้ใช้ได้
- ระบบต้องรองรับการลบบัญชีของผู้ใช้ได้
- ระบบต้องสามารถจัดการการเข้าสู่ระบบและการออกจากระบบของผู้ใช้ได้
- ระบบต้องสามารถตรวจสอบความถูกต้องของอีเมลผู้ใช้และรหัสผ่านได้ เมื่อผู้ใช้ทำการเข้าสู่ระบบ
- ระบบต้องสามารถยืนยันตัวตนผู้ใช้ผ่านระบบ OTP ทางเบอร์โทรศัพท์ของผู้ใช้ได้
- ผู้ใช้สามารถรายงานปัญหาต่อ Admin ได้
- ผู้ใช้สามารถดำเนินการซื้อตัวเครื่องบิน และดำเนินการจ่ายเงินได้
- ผู้ใช้สามารถจองได้หลายเที่ยวบิน

2. ส่วนการจัดการตัวเครื่องบิน

- ระบบต้องสามารถจำแนกประเภทตัวเครื่องบินได้ ได้แก่ ตัวเครื่องบินของเที่ยวบินภายในประเทศ และ ตัวเครื่องบินของเที่ยวบินข้ามประเทศ
- ระบบต้องสามารถจำแนกได้ว่าตัวเครื่องบินถูกซื้อโดยผู้ใด
- ระบบต้องสามารถจำแนกได้ว่าตัวเครื่องบินเป็นของเที่ยวบินใด
- ระบบต้องสามารถออกตั๋วโดยสารให้แก่ผู้โดยสารที่ชำระเงินแล้ว
- ระบบต้องสามารถเก็บข้อมูลการจองที่นั่งของตัวแต่ละใบ

3. ส่วนการจัดการการจองตัวเครื่องบิน

- ระบบต้องสามารถแสดงผลข้อมูลเที่ยวบิน ได้แก่ เที่ยวบิน วันและเวลาบิน ที่นั่ง ใต้อีเมล สนามบินต้นทาง-ปลายทาง อาคารผู้โดยสาร รหัสสุรกรรม ชื่อผู้โดยสาร สายการบิน และราคา
- ระบบต้องสามารถแสดงสถานะของสัมภาระได้

- ระบบต้องสามารถเลือกได้ว่า หากผู้ใช้งานทำการบินภายในประเทศ ระบบจะทำการเก็บข้อมูลเลขบัตรประจำตัวประชาชนแทน
- แต่ถ้าหากผู้ใช้งานทำการบินระหว่างประเทศ ระบบจะทำการเก็บข้อมูล หมายเลขหนังสือเดินทาง
- เมื่อผู้ใช้ทำการจองตัวเครื่องบิน ระบบจะส่งข้อมูลการจองว่าที่นั่งนี้ได้ถูกจองแล้วไปยัง สายการบิน และ ข้อมูลเที่ยวบิน

4. ส่วนการจัดการระบบ

- ระบบต้องรองรับการรายงานข้อความปัญหาจากผู้ใช้
- ระบบต้องรองรับให้ Admin สามารถตรวจสอบ ปัญหา หรือคำร้องเรียน จากผู้ใช้งานได้
- ระบบต้องอนุญาตให้ Admin ตรวจสอบและรับการติดต่อจากสายการบิน
- ระบบต้องอนุญาตให้ สายการบินสามารถติดต่อ Admin โดยการ ส่งข้อความได้

5. ส่วนการจัดการทำธุรกรรม

- ระบบต้องสามารถแสดงข้อมูลต่างๆเกี่ยวกับการทำธุรกรรม ได้แก่ หมายเลขธุรกรรม สถานะธุรกรรม วันเวลาที่ทำการชำระเงิน วิธีชำระเงิน จำนวนเงินที่ทำการชำระ รวมทั้งอัตราแลกเปลี่ยนสกุลเงิน
- ระบบต้องสามารถเก็บข้อมูลประวัติการทำธุรกรรมของผู้ใช้ได้
- ระบบต้องสามารถจำแนกได้ว่าการทำธุรกรรมต่างๆ เป็นของตัวเครื่องบินใด
- ระบบต้องรองรับการขอคืนเงินและยกเลิกการจองตัว

6. ส่วนการจัดการสายการบินและเครื่องบิน

- ระบบต้องสามารถเก็บข้อมูลของสายการบินต่างๆ ได้
- ระบบต้องสามารถเก็บข้อมูลของเครื่องบินในสายการบินต่างๆ ได้
- ระบบต้องสามารถเก็บจำนวนที่นั่งในเครื่องบินต่างๆ ได้
- ระบบต้องรองรับให้สายการบินเพิ่ม แก้ไข และลบข้อมูลของเครื่องบินได้
- ระบบต้องรองรับการเก็บข้อมูลของหมายเลขจดทะเบียน ของเครื่องบินของเที่ยวบินต่าง ๆ

7. ส่วนการจัดการเที่ยวบินและที่นั่ง

- ระบบต้องรองรับการกำหนดเที่ยวบินที่ออกเดินทางจากสนามบิน และไปยังปลายทาง
- ระบบต้องสามารถจัดเก็บข้อมูลของแต่ละเที่ยวบิน ได้แก่ หมายเลขเที่ยวบิน สายการบิน ราคา วันและเวลาบิน
- ระบบต้องสามารถจำแนกประเภทเที่ยวบินได้ ได้แก่ เที่ยวบินตรง และ เที่ยวบินต่อเครื่อง
- ระบบต้องรองรับให้สายการบินเพิ่ม แก้ไข และลบข้อมูลของเที่ยวบินได้
- ระบบต้องสามารถจัดเก็บข้อมูลสายการบินของเที่ยวบินนั้นๆ ได้
- ระบบต้องสามารถจัดเก็บข้อมูลของเครื่องบินที่ใช้ในเที่ยวบินนั้นๆ ได้
- ระบบต้องสามารถจัดเก็บข้อมูลของที่นั่งในเที่ยวบินนั้นๆ ได้
- ระบบต้องสามารถจัดเก็บข้อมูลของ จำนวนผู้โดยสารที่สามารถบรรจุได้ เลขที่นั่ง และ รูปแบบที่นั่งของทุกเที่ยวบิน
- ระบบต้องสามารถจัดเก็บข้อมูลของประเภทของเที่ยวบิน ว่าเป็นเที่ยวบินตรง หรือเที่ยวบินต่อ ถ้าหากเป็นเที่ยวบินต่อเครื่อง ต้องทำการจัดเก็บเมืองที่แวะพัก และ เวลาที่ใช้ในการแวะพักด้วย

8. ส่วนการจัดการสนามบิน

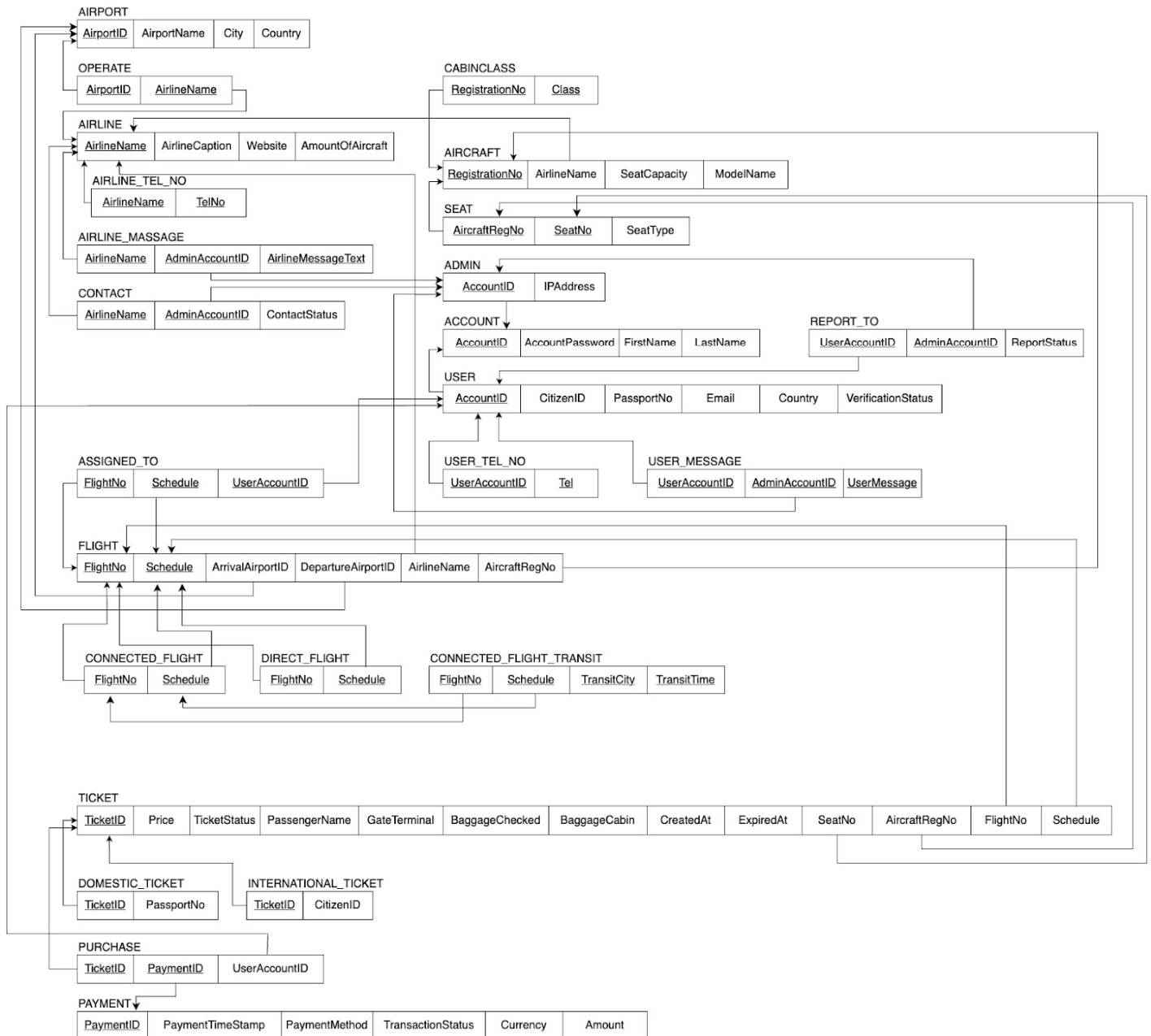
- ระบบต้องรองรับการเพิ่ม แก้ไข และลบข้อมูล สนามบิน
- ระบบต้องสามารถจัดเก็บข้อมูลของแต่ละสนามบิน ได้แก่ ไอดีสนามบิน ชื่อสนามบิน เมืองที่สนามบินนั้นตั้งอยู่ ประเทศที่สนามบินนั้นตั้งอยู่
- ระบบต้องสามารถจัดเก็บข้อมูลเที่ยวบินทั้งหมดที่จะบินขึ้นหรือลงจอดในสนามบินนั้นๆ ได้
- ระบบต้องสามารถจัดเก็บข้อมูลสายการบินทั้งหมดที่มีเที่ยวบินขึ้นบินและลงจอด ณ สนามบินนั้นๆ ได้

ER Diagram

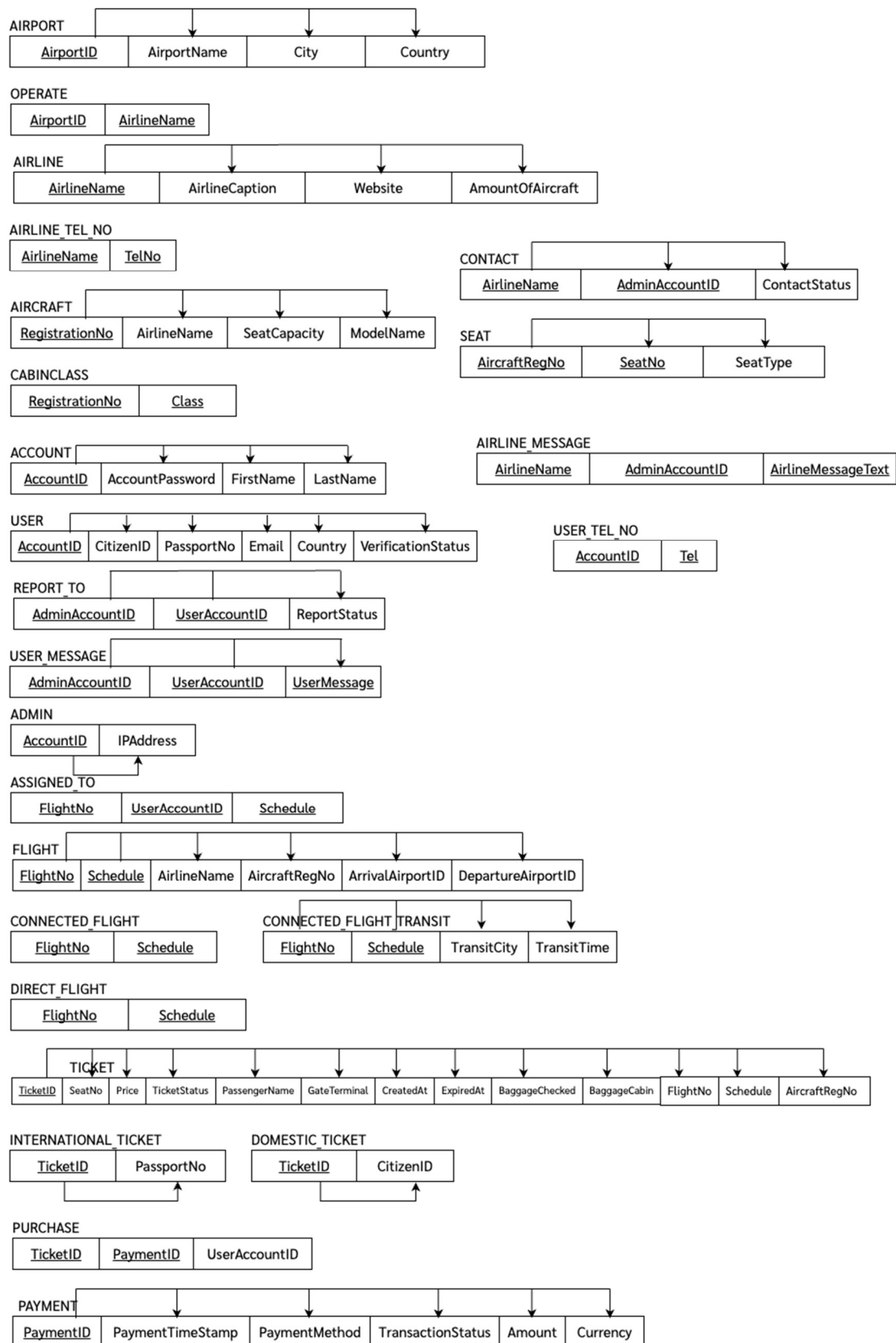
สามารถเข้าถึงดังต่อไปนี้เพื่อเข้าสู่ ER Diagram รูปเต็ม:

https://drive.google.com/drive/folders/1ioMLHNSy0KiC3_qLW_Omk9GAcF7g0h75?usp=drive_link

Schema Diagram



Normalization



Data Dictionary

Entity Types

No.	Name	Description
1	Account	Stores basic login and identity information for all users
2	Admin	A system administrator
3	App_User	A user of this website
4	Airport	An airport used as a departure or arrival point
5	Airline	An airline that operates flights
6	Aircraft	An airplane used for transportation
7	Flight	A scheduled air travel service operated by an airline
8	Connected_Flight	A connecting flight between two or more flights
9	Direct_Flight	A non-stop flight without layover
10	Seat	A passenger seat on an aircraft
11	Ticket	A travel document for a flight
12	Domestic_Ticket	A flight within the same country
13	International_Ticket	A flight between different countries
14	Payment	Payment information for ticket purchases

Entity Type Name: **ACCOUNT**

Attribute Name	Type	Descriptive Name	Valid Values	Allows Null
<u>AccountID</u>	VARCHAR(10)	Unique identifier for each account	Alphanumeric	No
AccountPassword	VARCHAR(100)	Password for the account	Any string	No
FirstName	VARCHAR(50)	First name of the user	Any string	No
LastName	VARCHAR(50)	Last name of the user	Any string	No

Entity Type Name: **ADMIN**

Attribute Name	Type	Descriptive Name	Valid Values	Allows Null
<u>AccountID</u>	VARCHAR(10)	Link to ACCOUNT	Alphanumeric	No
IPAddress	VARCHAR(45)	IP address of the admin	Valid IPv4 or IPv6 format	No

Entity Type Name: **APP_USER**

Attribute Name	Type	Descriptive Name	Valid Values	Allows Null
<u>AccountID</u>	VARCHAR(10)	Link to ACCOUNT	Alphanumeric	No
CitizenID	VARCHAR(20)	National ID	Alphanumeric	Yes
PassportNo	VARCHAR(20)	Passport number	Alphanumeric	Yes
Email	VARCHAR(100)	Email address	Valid email	Yes
VerificationStatus	BOOLEAN	Verification status	TRUE/FALSE	Yes

Country	VARCHAR(50)	Country of the user	Country name	No
---------	-------------	---------------------	--------------	----

Entity Type Name: **AIRPORT**

Attribute Name	Type	Descriptive Name	Valid Values	Allows Null
<u>AirportID</u>	CHAR(3)	Unique airport code	IATA 3-letter code	No
AirportName	VARCHAR(100)	Name of the airport	Any string	Yes
City	VARCHAR(50)	City of the airport	Any string	Yes
Country	VARCHAR(50)	Country of the airport	Any string	Yes

Entity Type Name: **AIRLINE**

Attribute Name	Type	Descriptive Name	Valid Values	Allows Null
<u>AirlineName</u>	VARCHAR(100)	Name of the airline	Any string	No
AirlineCaption	VARCHAR(100)	Slogan	Any string	Yes
Website	VARCHAR(100)	Website URL	Valid URL	Yes
AmountOfAircraft	INT	Number of aircraft	Integer ≥ 0	Yes

Entity Type Name: **AIRCRAFT**

Attribute Name	Type	Descriptive Name	Valid Values	Allows Null
<u>RegistrationNo</u>	VARCHAR(20)	Aircraft registration	Alphanumeric	No
AirlineName	VARCHAR(100)	Belongs to airline	Airline name	Yes
SeatCapacity	INT	Seat count	Integer > 0	No
ModelName	VARCHAR(50)	Aircraft model	Any string	Yes

Entity Type Name: FLIGHT

Attribute Name	Type	Descriptive Name	Valid Values	Allows Null
<u>FlightNo</u>	VARCHAR(10)	Unique flight ID	Alphanumeric	No
<u>Schedule</u>	TIMESTAMP	Scheduled date/time	Date and time	No
ArrivalAirportID	CHAR(3)	Destination airport	IATA code	No
DepartureAirportID	CHAR(3)	Departure airport	IATA code	No
AirlineName	VARCHAR(100)	Operated by airline	Airline name	No
AircraftRegNo	VARCHAR(20)	Assigned aircraft	Registration number	No

Entity Type Name: CONNECTED_FLIGHT

Attribute Name	Type	Descriptive Name	Valid Values	Allows Null
<u>FlightNo</u>	VARCHAR(10)	Linked flight	Alphanumeric	No
<u>Schedule</u>	TIMESTAMP	Flight schedule	Date/time	No

Entity Type Name: DIRECT_FLIGHT

Attribute Name	Type	Descriptive Name	Valid Values	Allows Null
<u>FlightNo</u>	VARCHAR(10)	Linked flight	Alphanumeric	No
<u>Schedule</u>	TIMESTAMP	Flight schedule	Date/time	No

Entity Type Name: **SEAT**

Attribute Name	Type	Descriptive Name	Valid Values	Allows Null
<u>AircraftRegNo</u>	VARCHAR(10)	Aircraft ID	Registration number	No
<u>SeatNo</u>	VARCHAR(10)	Seat number	Alphanumeric	No
SeatType	VARCHAR(20)	Seat type	'Economy', 'Business', 'First Class'	Yes

Entity Type Name: **TICKET**

Attribute Name	Type	Descriptive Name	Valid Values	Allows Null
<u>TicketID</u>	VARCHAR(10)	Ticket ID	Alphanumeric	No
PassengerName	VARCHAR(100)	Name of passenger	Any string	No
SeatNo	VARCHAR(10)	Assigned seat	Alphanumeric	No
Schedule	TIMESTAMP	Flight schedule	Date/time	Yes
FlightNo	VARCHAR(10)	Linked flight	Alphanumeric	No
Price	DECIMAL(10,2)	Ticket price	Decimal ≥ 0	No
TicketStatus	VARCHAR(20)	Status of ticket	'Confirmed', 'Cancelled', 'Pending'	Yes
CheckedBaggage	INT	Checked baggage (kg)	Integer ≥ 0	Yes
CabinBaggage	INT	Cabin baggage (kg)	Integer ≥ 0	Yes

GateTerminal	VARCHAR(10)	Boarding gate	Any string	Yes
CreatedAt	TIMESTAMP	Ticket created date	Date/time	Yes
ExpiredAt	TIMESTAMP	Ticket expiration date	Date/time	Yes
RegistrationNo	VARCHAR(20)	Linked aircraft	Alphanumeric	No

Entity Type Name: **DOMESTIC_TICKET**

Attribute Name	Type	Descriptive Name	Valid Values	Allows Null
<u>TicketID</u>	VARCHAR(10)	Linked ticket	Alphanumeric	No
CitizenID	VARCHAR(20)	National ID of passenger	Alphanumeric	No

Entity Type Name: **INTERNATIONAL_TICKET**

Attribute Name	Type	Descriptive Name	Valid Values	Allows Null
<u>TicketID</u>	VARCHAR(10)	Linked ticket	Alphanumeric	No
PassportNo	VARCHAR(20)	Passport number of passenger	Alphanumeric	No

Entity Type Name: **PAYMENT**

Attribute Name	Type	Descriptive Name	Valid Values	Allows Null
<u>PaymentID</u>	VARCHAR(10)	Payment ID	Alphanumeric	No
Amount	DECIMAL(10,2)	Payment amount	Decimal > 0	No

Currency	VARCHAR(10)	Currency code	E.g., USD, EUR	Yes
PaymentTimeStamp	TIMESTAMP	Payment time	Date/time	Yes
PaymentMethod	VARCHAR(50)	Method used	'Credit/Debit Card','eBanking','PayPal'	Yes
TransactionStatus	VARCHAR(20)	Transaction status	'Success', 'Pending', 'Failed'	Yes

Relationship Types

No.	Name	Description
1	Operate	Which airlines operate at which airports.
2	Manage	Which airline manages which flight.
3	Depart_From	From which airport a specific flight departs.
4	Arrive_At	At which airport a specific flight arrives.
5	Contain	The seats that are included within a particular flight.
6	Own	Which airline owns which aircraft.
7	Feature	What special features are offered on a flight, potentially linked to a specific aircraft type.
8	Assigned_To	Which user is assigned to which flight (e.g., after purchasing a ticket).
9	Belong_To	Which ticket is purchased for travel on which flight.
10	Purchase	Which user purchases which ticket.

11	Occupy	Which user is assigned to which seat (e.g., after checking in).
12	Contact	Which messages (from users) are associated with which airline, including a processing status.
13	Report_To	Which user submits which complaint message.

Relationship Type Name: **CONTACT**

Attribute Name	Type	Descriptive Name	Valid Values	Allows Null
ContactStatus	VARCHAR(20)	Status of message between Admin and Airline	'Open', 'InProgress', 'Resolved'	No

Relationship Type Name: **REPORTED_TO**

Attribute Name	Type	Descriptive Name	Valid Values	Allows Null
ReportStatus	VARCHAR(20)	Status of message between User and Admin	'Open', 'InProgress', 'Resolved'	No

DDL IN APPLICATION: INSERT, DELETE, UPDATE

INSERT

Scenario 1: มีผู้ใช้เข้ามาจองตั๋ว (ticket)

```
INSERT INTO ticket
(TicketID, PassengerName, SeatNo, Schedule, FlightNo, Price, TicketStatus, CheckedBaggage, CabinBaggage, GateTerminal, CreatedAt, ExpiredAt, RegistrationNo)
VALUES ('T003', 'Jane Nee', '13A', '2025-05-01
10:00:00', 'FS100', 500.00, 'Confirmed', 1, 1, 'A1', '2025-04-01
12:00:00', '2025-05-01 09:00:00', 'HS-FS001');
```

	ticketid [PK] character varying (10)	passengername character varying (100)	seatno character varying (10)	schedule timestamp without time zone	flightno character varying (10)	price numeric (10,2)	ticketstatus character varying (20)	checkedbaggage integer
1	T001	Jane Smith	12A	2025-05-01 10:00:00	FS100	500.00	Confirmed	
2	T002	John Doe	55B	2025-06-01 11:00:00	FJ200	2500.00	Confirmed	
3	T003	Jane Nee	13A	2025-05-01 10:00:00	FS100	500.00	Confirmed	

Scenario 2: ticketID เดียวกัน

```
INSERT INTO ticket
(TicketID, PassengerName, SeatNo, Schedule, FlightNo, Price, TicketStatus, CheckedBaggage, CabinBaggage, GateTerminal, CreatedAt, ExpiredAt, RegistrationNo)
VALUES ('T003', 'John Nee', '33A', '2025-05-01
10:00:00', 'FS100', 500.00, 'Confirmed', 1, 1, 'A1', '2025-04-01
12:00:00', '2025-05-01 09:00:00', 'HS-FS001');
```

ERROR: Key (ticketid)=(T003) already exists.duplicate key value violates unique constraint "ticket_pkey"

ERROR: duplicate key value violates unique constraint "ticket_pkey"

SQL state: 23505

Detail: Key (ticketid)=(T003) already exists.

Scenario 3: เพิ่ม user

```
INSERT INTO APP_USER
(AccountID, CitizenID, PassportNo, Email, VerificationStatus, Country)
VALUES
('A004', 'CID12334', 'P12345', 'use1r@example.com', TRUE, 'United States');
```

	accountid [PK] character varying (10)	citizenid character varying (20)	passportno character varying (20)	email character varying (100)	verificationstatus boolean	country character varying (50)
1		1234567890123	P9193939	jane@example.com	true	Thailand
2	A001	1030204949495	P1234567	john@example.com	true	Japan
3	A004	CID12334	P12345	use1r@example.com	true	United States

Scenario 4: violate unique constraint (citizenID)

```
INSERT INTO APP_USER
(AccountID,CitizenID,PassportNo,Email,VerificationStat
us,Country)
VALUES
('A005','CID12334','P67891','use2r@example.com',TRUE,'
United States');
```

ERROR: Key (citizenid)=(CID12334) already exists.duplicate key value violates unique constraint "app_user_citizenid_key"

ERROR: duplicate key value violates unique constraint "app_user_citizenid_key"

SQL state: 23505

Detail: Key (citizenid)=(CID12334) already exists.

DELETE

Scenario: ลบข้อมูลของตั๋ว (ticket) ตามticket_id (ticket_idนี้มีการอ้างอิงในตารางอื่น)

```
DELETE FROM ticket
WHERE TicketID = 'T001';
```

ERROR: Key (ticketid)=(T001) is still referenced from table "purchase".update or delete on table "ticket" violates foreign key constraint "purchase_ticketid_fkey" on table "purchase"

ERROR: update or delete on table "ticket" violates foreign key constraint "purchase_ticketid_fkey" on table "purchase"




SQL state: 23503

Detail: Key (ticketid)=(T001) is still referenced from table "purchase".

UPDATE

Scenario: updateข้อมูลที่นั่ง (seat) ของตั๋ว (ticket) ตามseatno (seattype ไม่มีจริง)

```
UPDATE seat
SET seattype = 'lala'
WHERE seatno = '12A';
```

	aircraftregno [PK] character varying (10) 	seatno [PK] character varying (10) 	seattype character varying (20) 
1	HS-FS001	12A	Economy
2	JA123	55B	Business

ERROR: Failing row contains (HS-FS001, 12A, lala).new row for relation "seat" violates check constraint "seat_seattype_check"

ERROR: new row for relation "seat" violates check constraint "seat_seattype_check"

SQL state: 23514

Detail: Failing row contains (HS-FS001, 12A, lala).

Indexing

เนื่องจากระบบ FlyWithSigma นั้นเป็นระบบที่จะมีการเรียกดูข้อมูลของเที่ยวบินบ่อยครั้งเป็นอย่างมาก โดยแต่ละครั้งอาจจำเป็นต้องค้นหาข้อมูลด้วยเงื่อนไขต่าง ๆ โดยเฉพาะอย่างยิ่งการค้นหาด้วยหมายเลขเที่ยวบิน (FlightNo) ด้วยสนามบินขาเข้า/ขาออก และด้วยตารางเที่ยวบิน

ดังนั้น ด้วยเหตุผลข้างต้น ระบบ FlyWithSigma จึงจำเป็นต้องมีส่วนที่ช่วยให้การเข้าถึงข้อมูลเที่ยวบินเหล่านี้โดยไม่จำเป็นต้องทำการสแกนทั้งตาราง โดยในส่วนของการทำงาน Indexing จะใช้ Relation FLIGHT ควบคู่กับ Attributes ดังต่อไปนี้ FlightNo, Departure Airport, Arrival Airport และ Schedule (หรือหมายถึง วันเวลา - Date/DepartureTime) และในการทำ Indexing นั้นจะใช้โครงสร้าง Index (Index Structure) แบบ B+ Tree เนื่องจากเป็นโครงสร้างที่ช่วยให้สามารถค้นหาข้อมูลแบบช่วงได้ (range queries) สามารถค้นหาจากราก (root) ไปยังใบ (leaf) ได้อย่างรวดเร็วภายใต้เวลาระดับลอการิทึม อีกทั้งใบของ B+ Tree ยังเชื่อมต่อกันทำให้เข้าถึงข้อมูลถัดไปได้อย่างรวดเร็ว

อย่างไรก็ดี เพื่อความสะดวกในการค้นหา AIRPORT จาก CITY และ COUNTRY จึงมีการเพิ่ม Indexing ให้กับ AIRPORT ด้วย Attributes City และ Country โดยเสนอแนวทางการจัดการดังนี้

PostgreSQL สำหรับสร้าง B+ Tree Indexes:

```
-- FLIGHT Table
CREATE INDEX idx_flightno
ON FLIGHT(FlightNo);

CREATE INDEX idx_schedule
ON FLIGHT(Schedule);

CREATE INDEX idx_arrival_airport_id
ON FLIGHT(ArrivalAirportID);

CREATE INDEX idx_departure_airport_id
ON FLIGHT(DepartureAirportID);

CREATE INDEX idx_airport_city_country
ON AIRPORT(City, Country);
```

The image shows a PostgreSQL query execution plan for a Seq Scan on the flight_no_index. The plan is titled 'Seq Scan on flight_no_index'. It includes a description of the Seq Scan Node: 'finds relevant records by sequentially scanning the input record set. When reading from a table, Seq Scans (unlike Index Scans) perform a single read operation (only the table is read)'. The plan is divided into tabs: General, IO & Buffers, Output, Workers, and Misc. The General tab is selected, showing the following details: Timing: 0.016ms | 47%, Rows: 0 (Planned: 1) | ↑ over estimated, *Rows Removed by Filter: 1 | >99% | ⓘ, and Cost: 12.6 (Total: 12.6).

The image shows a PostgreSQL query execution plan for a Seq Scan on the flight table. The plan is titled 'Seq Scan on flight'. It includes a description of the Seq Scan Node: 'finds relevant records by sequentially scanning the input record set. When reading from a table, Seq Scans (unlike Index Scans) perform a single read operation (only the table is read)'. The plan is divided into tabs: General, IO & Buffers, Output, Workers, and Misc. The General tab is selected, showing the following details: Timing: 0.011ms | 44%, Rows: 0 (Planned: 1) | ↑ over estimated, *Rows Removed by Filter: 1 | >99% | ⓘ, and Cost: 1.01 (Total: 1.01).

Stored Procedures

Stored Procedure 1

Scenario:

สร้างการจองตั๋วเครื่องบินใหม่และการบันทึกการชำระเงินในสถานะ "Pending"

Query:

```
CREATE OR REPLACE PROCEDURE BookTicket (  
    IN p_UserID VARCHAR(10),  
    IN p_FlightNo VARCHAR(10),  
    IN p_SeatNo VARCHAR(10),  
    IN p_Schedule TIMESTAMP,  
    IN p_PassengerName VARCHAR(100),  
    IN p_CheckedBaggage INT,  
    IN p_CabinBaggage INT,  
    IN p_GateTerminal VARCHAR(10),  
    IN p_Price DECIMAL(10,2),  
    IN p_RegistrationNo VARCHAR(20),  
    IN p_Currency VARCHAR(10),  
    IN p_PaymentMethod VARCHAR(50)  
)  
LANGUAGE plpgsql  
AS $$  
DECLARE  
    newTicketID VARCHAR(10);  
    isUnique BOOLEAN DEFAULT FALSE;  
    attempts INT DEFAULT 0;  
    expTime TIMESTAMP;  
    newPaymentID VARCHAR(10);  
BEGIN  
    expTime := CURRENT_TIMESTAMP + INTERVAL '24 HOURS';  
  
    WHILE NOT isUnique AND attempts < 100 LOOP  
        newTicketID := 'T' || LPAD(FLOOR(RANDOM() * 1000000)::TEXT, 6, '0');  
        IF NOT EXISTS (SELECT 1 FROM TICKET WHERE TicketID = newTicketID)  
    THEN  
            isUnique := TRUE;  
        END IF;  
        attempts := attempts + 1;  
    END LOOP;  
  
    IF NOT isUnique THEN  
        RAISE EXCEPTION 'Cannot generate a unique TicketID. Try again.';  
    END IF;  
  
    newPaymentID := 'P' || LPAD(FLOOR(RANDOM() * 1000000)::TEXT, 6, '0');
```

```

BEGIN

    INSERT INTO TICKET (
        TicketID, PassengerName, SeatNo, Schedule, FlightNo, Price,
        CheckedBaggage, CabinBaggage, GateTerminal, ExpiredAt, RegistrationNo
    ) VALUES (
        newTicketID, p_PassengerName, p_SeatNo, p_Schedule, p_FlightNo,
        p_Price, p_CheckedBaggage, p_CabinBaggage, p_GateTerminal, expTime,
        p_RegistrationNo
    );

    INSERT INTO PAYMENT (PaymentID, Amount, Currency, PaymentTimeStamp,
        PaymentMethod, TransactionStatus)
    VALUES (newPaymentID, p_Price, p_Currency, NULL, p_PaymentMethod,
        'Pending');

    INSERT INTO PURCHASE (UserAccountID, PaymentID, TicketID)
    VALUES (p_UserID, newPaymentID, newTicketID);

EXCEPTION
    WHEN OTHERS THEN
        RAISE NOTICE 'Error occurred. Rolling back...';
        RAISE;

END;
END;
$$;

```

Example Usage:

```

CALL BookTicket (
    'A005',
    'FS100',
    '5F',
    '2025-05-01 10:00:00',
    'Chatrin Verygood',
    20,
    7,
    'A2',
    7800.89,
    'HS-FS001',
    'THB',
    'PayPal'
);

SELECT * FROM TICKET;

```


Result:

	ticketid [PK] character varying (10) ↗	passengername character varying (100) ↗	seatno character varying (10) ↗	schedule timestamp without time zone ↗	flightno character varying (10) ↗	price numeric (10,2) ↗	ticketstatus character varying (20) ↗	checkedbaggage integer ↗	cabinbaggage integer ↗
1	T001	Jane Smith	12A	2025-05-01 10:00:00	FS100	500.00	Confirmed	1	1
2	T002	John Doe	55B	2025-06-01 11:00:00	FJ200	2500.00	Confirmed	1	1
3	T927531	Chatrin Verygood	5F	2025-05-01 10:00:00	FS100	7800.89	Pending	20	7

Stored Procedure 2

Scenario:

อัปเดตสถานะการชำระเงินเมื่อผู้ใช้ชำระเงินสำหรับการจองตั๋ว

Query:

```
CREATE OR REPLACE PROCEDURE MakePayment(  
    IN p_UserID VARCHAR(10),  
    IN p_TicketID VARCHAR(10),  
    IN p_Amount DECIMAL(10,2),  
    IN p_Currency VARCHAR(10),  
    IN p_Method VARCHAR(50)  
)  
LANGUAGE plpgsql  
AS  
$$  
DECLARE  
    existingPaymentID VARCHAR(10);  
    expectedAmount DECIMAL(10,2);  
    currentStatus VARCHAR(20);  
BEGIN  
  
    SELECT PaymentID INTO existingPaymentID  
    FROM PURCHASE  
    WHERE TicketID = p_TicketID AND UserAccountID = p_UserID;  
  
    IF existingPaymentID IS NULL THEN  
        RAISE EXCEPTION 'No payment record found for TicketID % and UserID  
%', p_TicketID, p_UserID;  
    END IF;  
  
    UPDATE PAYMENT  
    SET  
        Amount = p_Amount,  
        Currency = p_Currency,  
        PaymentMethod = p_Method,  
        PaymentTimeStamp = CURRENT_TIMESTAMP,  
        TransactionStatus = 'Success'  
    WHERE PaymentID = existingPaymentID;  
  
    UPDATE TICKET
```

```

SET TicketStatus = 'Confirmed'
WHERE TicketID = p_TicketID;
END;
$$;

```

Example Usage:

```

CALL MakePayment (
    'A005',
    'T927531',
    7800.89,
    'THB',
    'PayPal'
);

SELECT * FROM PAYMENT;

```

Result:

	paymentid [PK] character varying (10)	amount numeric (10,2)	currency character varying (10)	paymenttimestamp timestamp without time zone	paymentmethod character varying (50)	transactionstatus character varying (20)
1	P001	500.00	THB	2025-04-01 12:05:00	Credit Card	Success
2	P002	2500.00	THB	2025-04-01 13:00:00	Credit Card	Pending
3	P881378	7800.89	THB	2025-04-18 16:14:59.58968	PayPal	Success

Stored Functions

Store Function 1

Scenario:

➤ `get_avail_seats(func_flightNo VARCHAR, func_Schedule TIMESTAMP)`

เมื่อต้องการคำนวณจำนวนที่นั่งว่างที่เหลืออยู่ในเที่ยวบินแต่ละเที่ยว ณ วันที่และเวลาที่กำหนด ตัวอย่างหนึ่งของการใช้งานคือในระบบจองตั๋วออนไลน์ของสายการบิน เมื่อผู้โดยสารเลือกเส้นทางและวันที่เดินทาง ระบบจะต้องแสดงรายการเที่ยวบินทั้งหมดในวันนั้นพร้อมจำนวนที่นั่งว่างที่เหลืออยู่ เพื่อให้ผู้โดยสารสามารถตัดสินใจเลือกเที่ยวบินที่เหมาะสมได้ ระบบจะเรียกใช้ฟังก์ชันนี้โดยส่งหมายเลขเที่ยวบินและเวลาที่กำหนดเข้าไป จากนั้นฟังก์ชันจะคำนวณจากจำนวนที่นั่งทั้งหมดของเครื่องบินที่ใช้ในเที่ยวบินนั้น แล้วลบด้วยจำนวนที่นั่งที่มีผู้จองไปแล้ว (โดยไม่นับตั๋วที่ถูกยกเลิก) นอกจากนี้ ฟังก์ชันยังสามารถถูกนำไปใช้ในสถานการณ์อื่น เช่น ใช้ในระบบหลังบ้านของสายการบินเพื่อติดตามสถานะของเที่ยวบินต่าง ๆ เพื่อวางแผนโปรโมชั่นหรือปรับตารางบินในอนาคต หากเที่ยวบินใดมีจำนวนที่นั่งเหลือมากผิดปกติ ฟังก์ชันนี้จึงเป็นองค์ประกอบสำคัญที่ช่วยให้ระบบสามารถตัดสินใจและให้ข้อมูลแบบ real-time ได้อย่างมีประสิทธิภาพ

Store Function Code:

```
CREATE OR REPLACE FUNCTION get_avail_seats(func_flightNo VARCHAR, func_Schedule
TIMESTAMP)
RETURNS INT
LANGUAGE plpgsql
AS $$
    DECLARE total_seats INT;
    DECLARE booked_seats INT;
BEGIN
    SELECT SeatCapacity
    INTO total_seats
    FROM AIRCRAFT A
    JOIN FLIGHT F ON A.RegistrationNo = F.AircraftRegNo
    WHERE F.FlightNo = func_flightNo AND F.Schedule = func_Schedule;

    SELECT COUNT(*)
    INTO booked_seats
    FROM TICKET t
    WHERE t.FlightNo = func_flightNo AND t.Schedule = func_Schedule AND
    t.ticketstatus != 'Cancelled';

    RETURN COALESCE(total_seats - booked_seats,0);
END;
$$;
```

```

23 -- Testing Query (Should show how many seat that are available)
24 v SELECT f.flightNo,f.Schedule,get_avail_seats(f.flightNo,f.Schedule) as AvailableSeats
25 FROM flight f;
26

```

Data Output Messages Notifications

	flightno [PK] character varying (10)	schedule [PK] timestamp without time zone	availableseats integer
1	FS100	2025-05-01 10:00:00	179
2	SG102	2025-10-01 10:00:00	140

(ตัวอย่างการใช้งาน stored function get_avail_seats)

Store Function 2

Scenario:

➤ `cal_total_price(func_userAccountID VARCHAR)`

ใช้ในการคำนวณยอดใช้จ่ายรวมทั้งหมดของผู้ใช้งานแต่ละคน โดยอ้างอิงจากเลขบัญชีผู้ใช้ (UserAccountID) ที่ส่งเข้ามาเป็นพารามิเตอร์ การใช้งานของฟังก์ชันนี้มักจะเกิดขึ้นในระบบที่เกี่ยวข้องกับการติดตามยอดใช้จ่ายของผู้ใช้ เช่น ในระบบอีคอมเมิร์ซหรือแอปชำระเงินต่าง ๆ ตัวอย่างเช่น เมื่อผู้ใช้งานเข้าสู่หน้า "ประวัติการใช้จ่าย" หรือ "สรุปยอดรวม" ระบบอาจเรียกใช้ฟังก์ชันนี้เพื่อแสดงยอดเงินทั้งหมดที่ผู้ใช้นั้นเคยจ่ายไปโดยสมบูรณ์ โดยพิจารณาเฉพาะธุรกรรมที่มีสถานะว่า "Success" เท่านั้น ทำให้สามารถกรองธุรกรรมที่ล้มเหลวหรือถูกยกเลิกออกไปได้ ในทางปฏิบัติ การเรียกใช้ฟังก์ชันนี้อาจเกิดขึ้นได้ในหลายสถานการณ์ เช่น ฝ่ายบัญชีต้องการสร้างรายงานยอดใช้จ่ายรายเดือนของผู้ใช้แต่ละคน, หรือผู้ใช้งานเองต้องการตรวจสอบว่าตนเองใช้เงินในระบบไปเท่าใดในช่วงที่ผ่านมา ฟังก์ชันนี้จึงมีบทบาทสำคัญทั้งในเชิงผู้ใช้งานและในมุมมองของการจัดการระบบภายใน

Store Function Code:

```
CREATE OR REPLACE FUNCTION cal_total_price(func_userAccountID VARCHAR)
RETURNS DECIMAL(10,2)
LANGUAGE plpgsql
AS $$
DECLARE total_spent DECIMAL(10,2);
BEGIN
    SELECT SUM(p.Amount) INTO total_spent
    FROM PAYMENT p
    NATURAL JOIN PURCHASE pu
    WHERE p.TransactionStatus = 'Success'
    AND pu.UserAccountID = func_userAccountID;

    RETURN COALESCE(total_spent, 0);
END;
$$;
```

```
22 -- Testing Query (Should show All account id with their totalSpent)
23 ✓ SELECT accountid,cal_total_price(accountid) as totalSpent
24 FROM account;
```

Data Output Messages Notifications



	accountid [PK] character varying (10)	totalspent numeric
1	A001	0
2	A002	90000.00

(ตัวอย่างการใช้งาน cal_total_price)

Triggers

Trigger 1

Scenario:

เมื่อมีการ INSERT ค่าใน Table Ticket จะมี Trigger Function ไว้ตรวจสอบว่า Flight นั้นเต็มรึยัง และ Flight นั้นบินหรือยัง ถ้าหากใน 2 เงื่อนไขนี้มีอันใดอันหนึ่งเป็นจริงจะมี Error Message ขึ้นมา และ ไม่อนุญาตให้ INSERT ข้อมูลนั้นเข้ามา

Query:

```
-- This will Work only if you run Stored Function before running this
CREATE OR REPLACE FUNCTION trg_prevent_invalid_booking()
RETURNS TRIGGER
LANGUAGE plpgsql
AS $$
DECLARE available INT;
BEGIN
    SELECT get_avail_seats(NEW.FlightNo,NEW.Schedule)
    INTO available;

    IF available <= 0 THEN
        RAISE EXCEPTION 'Flight is full';
    END IF;

    IF NEW.Schedule <= NOW() THEN
        RAISE EXCEPTION 'Flight is already departed';
    END IF;
    RETURN NEW;
END;
$$;

CREATE TRIGGER adding_new_ticket
BEFORE INSERT ON TICKET
FOR EACH ROW
EXECUTE PROCEDURE trg_prevent_invalid_booking();
```

Trigger 2

Scenario:

เมื่อมีการ UPDATE ข้อมูลใน Relation PURCHASE หาก User นั้นยังไม่ได้ยืนยันตัวตน หรือ Payment นั้นถูกไปแล้วจะไม่อนุญาตให้ UPDATE Payment นั้นอีกและจะมี Error Message แสดงผลขึ้นมา

Query:

```
CREATE OR REPLACE FUNCTION trg_user_purchase()
RETURNS TRIGGER
LANGUAGE plpgsql
AS $$
DECLARE
    verify BOOLEAN;
    existingPaymentID VARCHAR(10);
    expectedAmount DECIMAL(10,2);
    currentStatus VARCHAR(20);
BEGIN
    -- Check User Verification
    SELECT u.VerificationStatus INTO verify
    FROM app_user u
    WHERE NEW.UserAccountID = u.AccountID;

    IF NOT verify THEN
        RAISE EXCEPTION 'User is not verified';
    END IF;

    -- Check that if payment already paid
    SELECT TransactionStatus INTO currentStatus
    FROM PAYMENT
    WHERE PaymentID = NEW.paymentid;

    IF currentStatus = 'Success' THEN
        RAISE EXCEPTION 'Payment has already been completed.';
    END IF;

    RETURN NEW;
END;
$$;

CREATE TRIGGER check_user_purchase
BEFORE INSERT ON PURCHASE
FOR EACH ROW
EXECUTE PROCEDURE trg_user_purchase();
```

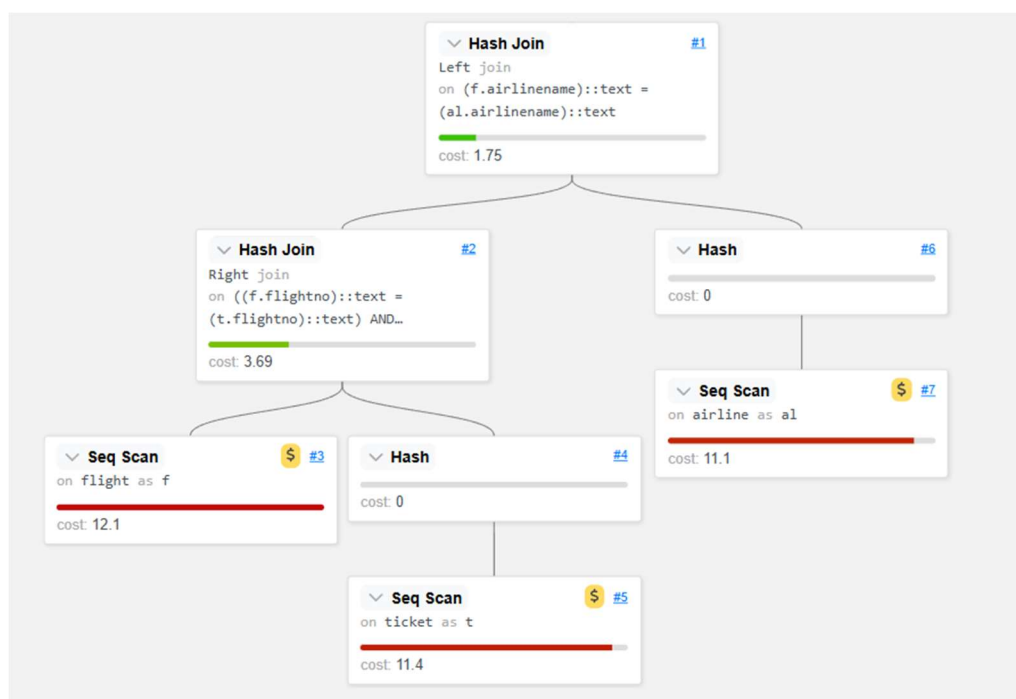

Execution Path

Scenario:

เมื่อต้องการดึงข้อมูลการจองตั๋วเครื่องบินจากระบบ

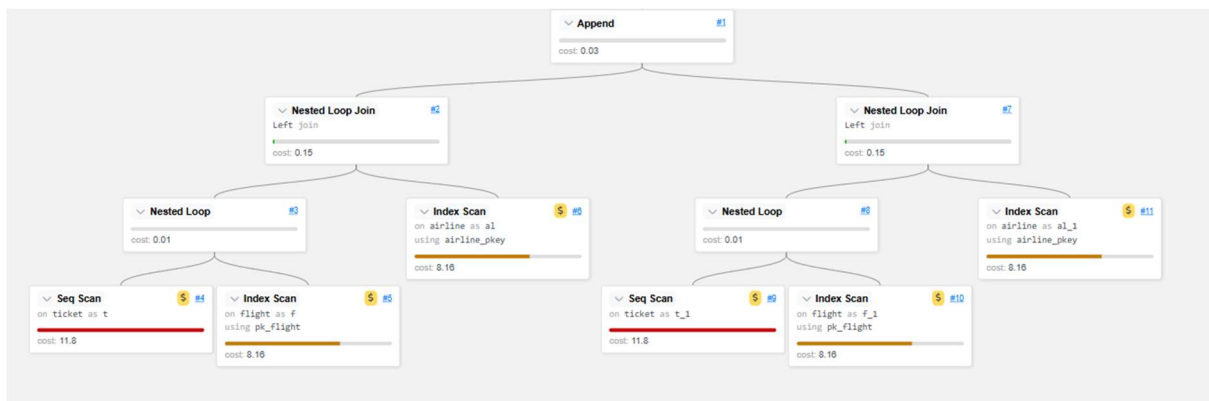
Query #1

```
SELECT
  t.TicketID,
  t.PassengerName,
  t.SeatNo,
  f.FlightNo,
  f.Schedule,
  al.AirlineName,
  al.AirlineCaption
FROM TICKET t
LEFT JOIN FLIGHT f ON t.FlightNo = f.FlightNo AND t.Schedule = f.Schedule
LEFT JOIN AIRLINE al ON al.AirlineName = f.AirlineName;
```



Query #2

```
(SELECT
  t.TicketID,
  t.PassengerName,
  t.SeatNo,
  f.FlightNo,
  f.Schedule,
  al.AirlineName,
  al.AirlineCaption
FROM TICKET t
LEFT JOIN FLIGHT f ON t.FlightNo = f.FlightNo AND t.Schedule = f.Schedule
LEFT JOIN AIRLINE al ON al.AirlineName = f.AirlineName
WHERE f.FlightNo = 'FS100')
UNION ALL
(SELECT
  t.TicketID,
  t.PassengerName,
  t.SeatNo,
  f.FlightNo,
  f.Schedule,
  al.AirlineName,
  al.AirlineCaption
FROM TICKET t
LEFT JOIN FLIGHT f ON t.FlightNo = f.FlightNo AND t.Schedule = f.Schedule
LEFT JOIN AIRLINE al ON al.AirlineName = f.AirlineName
WHERE f.FlightNo = 'FJ200');
```



Three Complex SQL Queries

Scenario 1:

ดึงรายชื่อผู้โดยสารที่ได้จองเที่ยวบินที่ได้รับการยืนยันอย่างน้อยหนึ่งเที่ยว ระหว่างวันที่ 1 ถึง 15 พฤษภาคม 2025 โดยให้แสดงข้อมูลดังนี้: หมายเลขบัญชี (Account ID), ชื่อจริง, นามสกุล, หมายเลขหนังสือเดินทาง, จำนวนตั๋วที่ได้รับการยืนยัน, หมายเลขเที่ยวบินที่ไม่ซ้ำกันซึ่งผู้โดยสารได้จอง พร้อมชื่อสายการบินของเที่ยวบินนั้น จัดเรียงผลลัพธ์ตามจำนวนตั๋วที่ยืนยันจากมากไปน้อย จากนั้นเรียงตามนามสกุลและชื่อจริงในลำดับจากน้อยไปมาก (A ถึง Z)

Query:

```
SELECT
    u.AccountID,
    a.FirstName,
    a.LastName,
    u.PassportNo,
    COUNT(t.TicketID) AS ConfirmedTicketCount,
    STRING_AGG(DISTINCT t.FlightNo || ' (' || f.AirlineName || ')', ', '
    ) AS FlightsFlown
FROM APP_USER u
JOIN ACCOUNT a ON u.AccountID = a.AccountID
JOIN PURCHASE p ON p.UserAccountID = u.AccountID
JOIN TICKET t ON t.TicketID = p.TicketID
JOIN FLIGHT f ON f.FlightNo = t.FlightNo AND f.Schedule = t.Schedule
WHERE
    t.TicketStatus = 'Confirmed'
    AND t.Schedule BETWEEN '2025-05-01' AND '2025-05-15'
GROUP BY u.AccountID, a.FirstName, a.LastName, u.PassportNo
HAVING COUNT(t.TicketID) >= 1
ORDER BY ConfirmedTicketCount DESC, a.LastName, a.FirstName;
```

Result:

	accountid character varying (10) 🔒	firstname character varying (50) 🔒	lastname character varying (50) 🔒	passportno character varying (20) 🔒	confirmedticketcount bigint 🔒	flightsflown text 🔒
1	A002	Jane	Smith	P9193939	1	FS100 (FlySigma)

Scenario 2:

ค้นหาเที่ยวบินทั้งหมดที่มีกำหนดการเดินทางระหว่างวันที่ 1 เมษายน ถึง 31 พฤษภาคม 2025 ซึ่งมีผู้ใช้งานจากอย่างน้อยหนึ่งประเทศ จดตัวไว้ โดยให้แสดงข้อมูลดังต่อไปนี้: หมายเลขเที่ยวบิน, ชื่อสายการบิน, วันและเวลาที่เที่ยวบินออกเดินทาง, จำนวนประเทศที่ไม่ซ้ำกันของผู้โดยสารที่จองเที่ยวบินนั้น, อีเมลของผู้โดยสารทั้งหมดที่ซื้อบัตรโดยสารสำหรับเที่ยวบินนั้น โดยเรียงลำดับข้อมูลจากน้อยไปมาก ตามลำดับ: จำนวนประเทศที่ไม่ซ้ำกัน, วันและเวลาของเที่ยวบิน, หมายเลขเที่ยวบิน, ชื่อสายการบิน

Query:

```
SELECT
    f.FlightNo,
    f.AirlineName,
    f.Schedule,
    COUNT(DISTINCT u.Country) AS DistinctCountries,
    STRING_AGG(u.Email, ', ') AS UserEmails
FROM
    FLIGHT f
JOIN
    TICKET t ON f.FlightNo = t.FlightNo
JOIN
    PURCHASE p ON t.TicketID = p.TicketID
JOIN
    APP_USER u ON p.UserAccountID = u.AccountID
WHERE
    f.Schedule BETWEEN '2025-04-01' AND '2025-05-31'
GROUP BY
    f.FlightNo, f.AirlineName, f.Schedule
HAVING
    COUNT(DISTINCT u.Country) >= 1
ORDER BY
    DistinctCountries, f.Schedule, f.FlightNo,
    f.AirlineName;
```

Result:

	flightno [PK] character varying (10)	airlinename character varying (100)	schedule [PK] timestamp without time zone	distinctcountries bigint	useremails text
1	FS100	FlySigma	2025-05-01 10:00:00	1	jane@example.com

Scenario 3:

แสดงรายการเที่ยวบินระหว่างประเทศทั้งหมด (ประเทศต้นทางและประเทศปลายทางไม่เหมือนกัน) ที่มีกำหนดการเดินทางระหว่างวันที่ 1 พฤษภาคม ถึง 15 มิถุนายน 2025 โดยให้แสดงข้อมูลดังนี้: หมายเลขเที่ยวบิน, ชื่อสายการบิน, สนามบินต้นทางและประเทศต้นทาง, สนามบินปลายทางและประเทศปลายทาง, จำนวนผู้โดยสารทั้งหมดในแต่ละเที่ยวบิน เรียงลำดับผลลัพธ์โดยเรียงจากจำนวนผู้โดยสารมากไปน้อย และตามด้วยหมายเลขเที่ยวบินจากน้อยไปมาก

Query:

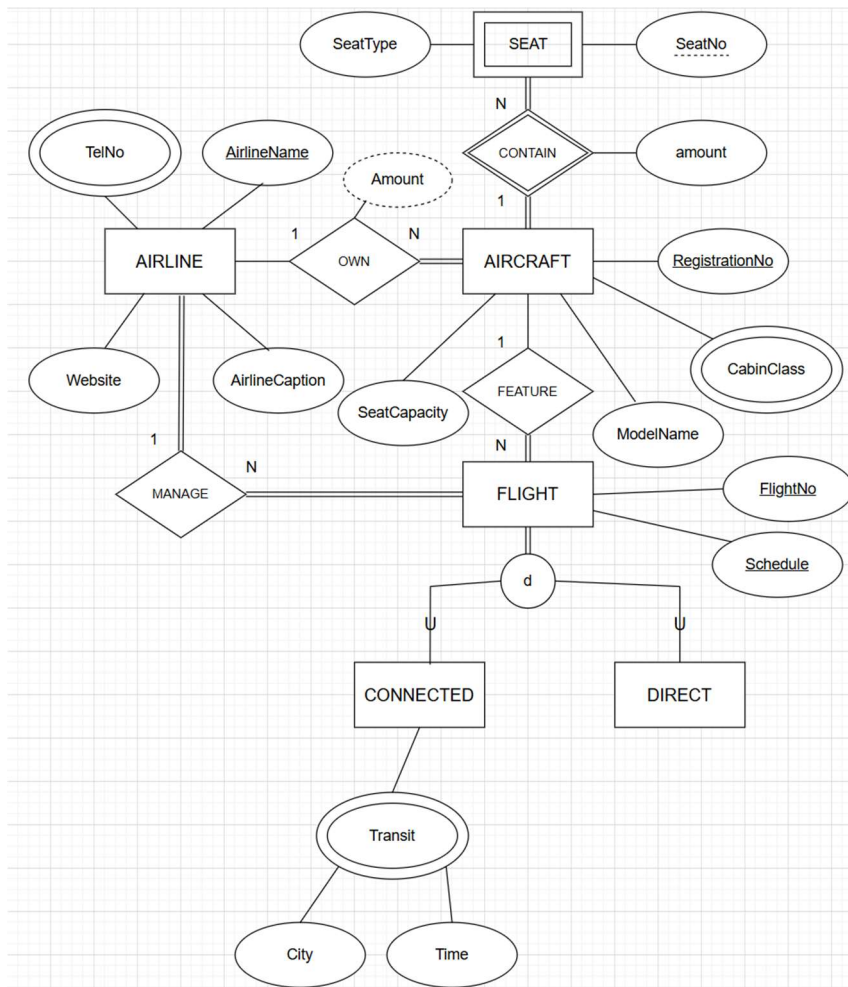
```
SELECT
    f.FlightNo,
    f.Schedule,
    al.AirlineName,
    dep.AirportName AS DepartureAirport,
    dep.Country AS DepartureCountry,
    arr.AirportName AS ArrivalAirport,
    arr.Country AS ArrivalCountry,
    COUNT(t.TicketID) AS TotalPassengers
FROM FLIGHT f
JOIN AIRPORT dep ON f.DepartureAirportID = dep.AirportID
JOIN AIRPORT arr ON f.ArrivalAirportID = arr.AirportID
JOIN AIRLINE al ON f.AirlineName = al.AirlineName
LEFT JOIN TICKET t ON f.FlightNo = t.FlightNo
AND f.Schedule = t.Schedule
WHERE
    f.Schedule BETWEEN '2025-05-01' AND '2025-06-15'
    AND dep.Country <> arr.Country
GROUP BY
    f.FlightNo,
    f.Schedule,
    al.AirlineName,
    dep.AirportName,
    dep.Country,
    arr.AirportName,
    arr.Country
ORDER BY
    TotalPassengers DESC,
    f.FlightNo;
```

Result:

	flightno character varying (10)	schedule timestamp without time zone	airlinename character varying (100)	departureairport character varying (100)	departurecountry character varying (50)	arrivalairport character varying (100)	arrivalcountry character varying (50)	totalpassengers bigint
1	J200	2025-06-01 11:00:00	FlyJapan	Haneda Airport	Japan	Suvarnabhumi Airport	Thailand	1

Document-based Design Schema

ER Diagram



สามารถแปลงเป็น document-based design schema ได้ดังต่อไปนี้

AIRCRAFT:

```
{
  "bsonType": "object",
  "required": ["RegistrationNo", "ModelName", "SeatCapacity",
    "CabinClass", "SeatAmount"],
  "properties": {
    "_id": { "bsonType": "objectId" },
    "RegistrationNo": { "bsonType": "string" },
    "ModelName": { "bsonType": "string" },
    "SeatCapacity": { "bsonType": "int" },
    "CabinClass": {
      "bsonType": "array",
      "minItems": 1,
      "maxItems": 3,
      "items": {
        "enum": ["ECONOMY", "BUSINESS", "FIRSTCLASS"]
      }
    }
  }
}
```

```

        },
        "uniqueItems": true
    },
    "SeatAmount": { "bsonType": "int" }
}

```

SEAT:

```

{
  "bsonType": "object",
  "required": ["AircraftId", "SeatType", "SeatNo"],
  "properties": {
    "_id": { "bsonType": "objectId" },

    "AircraftId": {
      "bsonType": "objectId",
      "description": "Reference to AIRCRAFT._id"
    },

    "SeatType": {
      "enum": ["ECONOMY", "BUSINESS", "FIRSTCLASS"]
    },

    "SeatNo": { "bsonType": "string" }
  }
}

```

AIRLINE:

```

{
  "bsonType": "object",
  "required": ["AirlineName", "TelNo"],
  "properties": {
    "_id": { "bsonType": "objectId" },
    "AirlineName": { "bsonType": "string" },
    "AirlineCaption": { "bsonType": "string" },
    "Website": { "bsonType": "string" },
    "TelNo": {
      "bsonType": "array",
      "items": { "bsonType": "string" }
    },

    "Aircrafts": {
      "bsonType": "array",
      "items": {
        "bsonType": "objectId"
      }
    }
  }
}

```

FLIGHT:

```
{
  "bsonType": "object",
  "required": ["FlightNo", "Schedule", "Type", "AircraftId", "AirlineId"],
  "properties": {
    "_id": { "bsonType": "objectId" },

    "FlightNo": { "bsonType": "string" },

    "Schedule": { "bsonType": "date" },

    "Type": { "enum": ["DIRECT", "CONNECTED"] },

    "AircraftId": {
      "bsonType": "objectId",
      "description": "Reference to AIRCRAFT._id"
    },

    "AirlineId": {
      "bsonType": "objectId",
      "description": "Reference to AIRLINE._id"
    },

    "Transit": {
      "bsonType": "object",
      "required": ["City", "Time"],
      "properties": {
        "City": { "bsonType": "string" },
        "Time": { "bsonType": "date" }
      }
    }
  }
}
```


MongoDB

ฟังก์ชันการค้นหาเที่ยวบินตรง ที่อยู่ในช่วงวันที่ 3 พฤษภาคม 2025 เวลา 00:00:00 ถึง 5 พฤษภาคม 2025 23:59:59 ของสายการบิน Qatar Airways

```
db.getCollection('flight').aggregate([
  {
    $match: {
      Type: 'DIRECT',
      Schedule: {
        $gte: ISODate(
          '2025-05-03T00:00:00.000Z'
        ),
        $lte: ISODate(
          '2025-05-06T23:59:59.000Z'
        )
      }
    }
  },
  {
    $lookup: {
      from: 'airline',
      localField: 'AirlineId',
      foreignField: '_id',
      as: 'airline'
    }
  },
  { $unwind: { path: '$airline' } },
  {
    $match: {
      'airline.AirlineName': 'Qatar Airways'
    }
  },
  {
    $lookup: {
      from: 'aircraft',
      localField: 'AircraftId',
      foreignField: '_id',
      as: 'aircraft'
    }
  },
  { $unwind: { path: '$aircraft' } },
  {
    $project: {
      _id: 0,
      FlightNo: 1,
      Type: 1,
      Schedule: 1,
      AirlineName: '$airline.AirlineName',
      ModelName: '$aircraft.ModelName',
      SeatCapacity: '$aircraft.SeatCapacity',
      SeatTypeAvailable: '$aircraft.CabinClass'
    }
  }
])
```

```
    }  
  }  
],  
{ maxTimeMS: 60000, allowDiskUse: true }  
);
```

หมายเหตุ

สามารถเข้าไปดูโค้ดและไฟล์ข้อมูล .csv ต่าง ๆ ได้ที่ <https://github.com/fforfaii/db-flywithsigma>