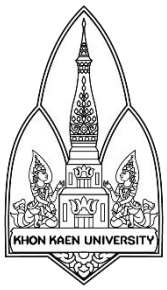


เอกสารรายงานฉบับสมบูรณ์

การสร้าง Multidimensional data model และแดชบอร์ดนำเสนอข้อมูลของฐานข้อมูล Chinook ผ่านมุมมองทางธุรกิจ

The Multidimensional data model and dashboard present the data of the Chinook database through a business perspective.

จัดทำโดย

653020209-3 นายธีรภัทร์ เพชรดง

653020215-8 นางสาวเพ็ญนภา แก้วมูลเมือง

อาจารย์ที่ปรึกษา

ดร.เปรม จันทร์สว่าง

รายงานนี้เป็นส่วนหนึ่งของรายวิชา SC663402 Data Warehouse and Big Data Analytics

คลังข้อมูลและการวิเคราะห์ข้อมูลขนาดใหญ่

สาขาวิชาสถิติและวิทยาการข้อมูล วิชาเอกสารสนเทศสถิติและวิทยาการข้อมูล คณะวิทยาศาสตร์ มหาวิทยาลัยขอนแก่น ปีการศึกษา 1/2567

ธีรภัทร์ เพชรดง และ เพ็ญนภา แก้วมูลเมือง 2567. การสร้าง Multidimensional data model และแดชบอร์ดนำเสนอ ข้อมูลของฐานข้อมูล Chinook ผ่านมุมมองทางธุรกิจ. Data Warehouse and Big Data Analytics คลังข้อมูลและการวิเคราะห์ข้อมูลขนาดใหญ่. สาขาวิชาสถิติและวิทยาการข้อมูล วิชาเอกสารสนเทศสถิติและวิทยาการข้อมูล คณะวิทยาศาสตร์ มหาวิทยาลัยขอนแก่น

อาจารย์ที่ปรึกษา : อ.ดร.เปรม จันทร์สว่าง

บทคัดย่อ

ในการสร้างโมเดลข้อมูลแบบมัลติไดเมนชัน (Multidimensional Data Model) และแดชบอร์ด เพื่อนำเสนอข้อมูลจากฐานข้อมูล Chinook ผ่านมุมมองทางธุรกิจ งานวิจัยนี้มีวัตถุประสงค์หลักสองประการ คือ 1) การสร้าง Multidimensional Data Model และออกแบบ Data Cube เพื่อจัดระเบียบข้อมูลในหลายมิติที่สามารถตอบสนองต่อคำถามทางธุรกิจที่หลากหลาย และเพื่อให้การวิเคราะห์ข้อมูลที่ซับซ้อนเป็นไปอย่างมีประสิทธิภาพ และ 2) การสร้าง Dashboard เพื่อนำเสนอผลลัพธ์ที่แสดงข้อมูลที่วิเคราะห์จาก Data Cube ในรูปแบบที่เข้าใจง่ายและรวดเร็ว ซึ่งจะช่วยให้ผู้บริหารสามารถเข้าถึงและใช้ข้อมูลสำคัญในการตัดสินใจ ทางธุรกิจได้อย่างมีประสิทธิภาพ การดำเนินงานในครั้งนี้ได้นำเสนอข้อมูล Chinook ผ่านมุมมองธุรกิจที่สามารถนำไปใช้ประกอบการตัดสินใจ โดยได้เรียกใช้ข้อมูลจากฐานข้อมูล Chinook เพื่อสร้างตารางสำหรับมุมมองธุรกิจจำนวน 10 มุมมอง ได้แก่ ยอดขายรวมของเพลงทั้งหมด ยอดขายรวมของเพลงในแต่ละช่วงเวลา ยอดขายตามประเภทของเพลง Top 5 ศิลปินที่มียอดขายสูงสุดในแต่ละปี Top 5 ประเทศที่มียอดซื้อสูงสุดในปีล่าสุด ยอดขายของพนักงานในแต่ละปี ยอดขายตามภูมิภาค แนวโน้มยอดขายเพลงในแต่ละแนวเพลง พนักงานที่มียอดขายสูงสุดในแต่ละปี และพนักงานที่มียอดขายสูงสุดโดยรวม จากการศึกษาพบว่าโมเดลข้อมูลที่ออกแบบในรูปแบบ Star Schema ช่วยเพิ่มประสิทธิภาพในการวิเคราะห์ข้อมูลเชิงลึก โดยมีการจัดระเบียบข้อมูลในลักษณะที่ช่วยให้ผู้ใช้สามารถเข้าถึงข้อมูลได้อย่างรวดเร็วและง่ายดาย การใช้ Fact Table และ Dimension Tables ที่ออกแบบมาอย่างเหมาะสมทำให้สามารถวิเคราะห์ข้อมูลในหลายมิติได้ รวมถึงการพิจารณาแนวโน้มในเชิงลึก เช่น การเปลี่ยนแปลงยอดขายตามช่วงเวลา และการเปรียบเทียบผลการขายระหว่างประเภทสินค้าและศิลปินที่แตกต่างกัน ในดเนของ Web Application ที่พัฒนาขึ้นได้รับการออกแบบให้สามารถนำเสนอข้อมูลในรูปแบบแดชบอร์ดที่เข้าใจง่าย มีการใช้กราฟและตารางเพื่อช่วยในการแสดงผลข้อมูลอย่างชัดเจน

การนำเสนอข้อมูลในรูปแบบนี้ทำให้ผู้บริหารสามารถใช้ข้อมูลในการวางแผนและตัดสินใจได้อย่างถูกต้องและรวดเร็ว ซึ่งช่วยให้การวิเคราะห์ข้อมูลสามารถดำเนินการได้ในหลายมิติ เช่น การติดตามประสิทธิภาพการขายของพนักงานในแต่ละปีและการวิเคราะห์ข้อมูลเชิงลึกของยอดขายตามภูมิภาค นอกจากนี้ ระบบยังมีฟังก์ชันการค้นหาและกรองข้อมูลที่ช่วยให้ผู้ใช้สามารถเข้าถึงข้อมูลเฉพาะเจาะจง ตามช่วงเวลาและกลุ่มลูกค้าได้อย่างมีประสิทธิภาพ การเชื่อมต่อกับฐานข้อมูล Multidimensional Data Model โดยตรงทำให้สามารถดึงข้อมูลออกมาแสดงผลได้แบบ Real-time และช่วยให้การตัดสินใจในเชิงกลยุทธ์ในอนาคตมีความแม่นยำและสามารถรองรับการเปลี่ยนแปลงที่เกิดขึ้นในธุรกิจได้อย่างทันท่วงที ด้วยเหตุนี้ แดชบอร์ดและโมเดลข้อมูลที่พัฒนาขึ้นจึงถือเป็นเครื่องมือสำคัญในการตัดสินใจเชิงกลยุทธ์ในอนาคตที่สามารถสร้างความได้เปรียบในการแข่งขันในตลาดที่เปลี่ยนแปลงอย่างรวดเร็วได้อย่างมีประสิทธิภาพ

กิตติกรรมประกาศ

ในการศึกษา ดำเนินการ สร้าง Multidimensional data model และแดชบอร์ดนำเสนอข้อมูลของฐานข้อมูล Chinook ผ่านมุมมองธุรกิจ ทางผู้จัดทำได้รับความอนุเคราะห์จากอาจารย์ที่ปรึกษา และคำแนะนำจากผู้มีความรู้ในเรื่องดังกล่าว รวมถึงได้รับความช่วยเหลือจากบุคคลหลายคน จึงอยากขอขอบคุณทุกท่านที่มีส่วนร่วมในการทำให้การดำเนินงานครั้งนี้สำเร็จลุล่วงไปได้ด้วยดี

ขอขอบคุณ อ.ดร.เปรม จันทร์สว่าง ที่เป็นที่ปรึกษาโครงงานนี้และให้คอยให้คำแนะนำแนวทางในการดำเนินงาน การคิด วิเคราะห์ การหาแนวทางในการแก้ไขปัญหาที่เกิดขึ้นในขณะที่กำลังดำเนินงาน จนทำให้สามารถแก้ไขปัญหาได้

ผู้จัดทำ

ธีรภัทร์ เพชรดง

เพ็ญนภา แก้วมูลเมือง

สารบัญ

บทที่ 1

ที่มาและความสำคัญ

1.1 ที่มาและความสำคัญ

ในปัจจุบันยุคที่ข้อมูลกลายเป็นทรัพยากรสำคัญในการตัดสินใจทางธุรกิจ การจัดการข้อมูลอย่างมีประสิทธิภาพจึงเป็นปัจจัยที่ส่งผลต่อความสำเร็จขององค์กรอย่างชัดเจน (Kimball, 2011). การเก็บข้อมูลในรูปแบบกระจัดกระจาย หรือการมีข้อมูลที่แยกอยู่ในหลายแหล่งที่ไม่เชื่อมโยงกัน สามารถทำให้เกิดความยุ่งยากในการรวบรวมข้อมูลที่จำเป็นสำหรับการวิเคราะห์และการตัดสินใจ ข้อมูลที่กระจัดกระจายทำให้การเชื่อมโยงและการวิเคราะห์ข้อมูลที่มีมิติหลายด้าน เช่น การเปรียบเทียบยอดขายตามช่วงเวลา, ประเภทสินค้า, และสถานที่ กลายเป็นเรื่องที่ซับซ้อนและใช้เวลานาน (Kimball, 2011). การใช้ฐานข้อมูลเชิงสัมพันธ์ (Relational Database) ที่จัดเก็บข้อมูลในรูปแบบตารางสามารถช่วยในการจัดระเบียบข้อมูลพื้นฐานได้ แต่ยังมีข้อจำกัดในการวิเคราะห์ข้อมูลที่ต้องการมุมมองหลายมิติ เนื่องจากตารางข้อมูลมักจะไม่เหมาะสำหรับการสรุปผลที่ซับซ้อนและการวิเคราะห์ที่มีมิติหลายด้าน (Häfner, 2005).

การสร้าง Multidimensional Data Model และ Data Cube จึงเป็นเครื่องมือที่สำคัญอย่างยิ่งในการจัดระเบียบข้อมูลและการนำเสนอข้อมูลในรูปแบบที่มีประสิทธิภาพสูงสุด (Häfner, 2005). Data Cube ช่วยให้สามารถจัดระเบียบข้อมูลในหลายมิติ เช่น เวลา, สถานที่, และประเภทสินค้า ทำให้การจัดการและการวิเคราะห์ข้อมูลที่ซับซ้อนสามารถทำได้อย่างสะดวกและมีประสิทธิภาพ (Kimball, 2011). โดยการใช้ Data Cube ผู้บริหารสามารถรวบรวมและจัดระเบียบข้อมูลจากหลายแหล่งข้อมูลที่แตกต่างกัน ซึ่งเป็นการผสานข้อมูลที่แยกออกจากกันให้เป็นข้อมูลที่สอดคล้องและเป็นระบบ ช่วยให้สามารถสร้างมุมมองที่หลากหลาย เช่น การวิเคราะห์ยอดขายตามปี, ประเภทเพลง, และศิลปิน (Kimball, 2011). การสร้างแดชบอร์ดที่เชื่อมโยงกับ Data Cube ช่วยในการแสดงข้อมูลในรูปแบบที่เข้าใจง่ายและรวดเร็ว โดยสามารถสร้างกราฟและรายงานที่มีความชัดเจนและครอบคลุม ซึ่งทำให้ผู้บริหารสามารถเข้าถึงข้อมูลที่สำคัญได้ทันที การเข้าถึงข้อมูลที่เป็นปัจจุบันและเชื่อถือได้ช่วยในการตัดสินใจทางธุรกิจโดยมีข้อมูลที่มีคุณภาพสูง สนับสนุนการวางแผนกลยุทธ์ที่แม่นยำและการติดตามประสิทธิภาพของธุรกิจอย่างมีประสิทธิภาพมากยิ่งขึ้น (Häfner, 2005). ข้อมูลที่จัดระเบียบในลักษณะนี้ทำให้การวิเคราะห์และการสรุปผลสามารถทำได้อย่างรวดเร็วและถูกต้อง ช่วยให้ผู้บริหารสามารถตัดสินใจได้อย่างมีข้อมูลรองรับและปรับกลยุทธ์ทางธุรกิจได้ตามความต้องการที่เกิดขึ้นในตลาดและการเปลี่ยนแปลงที่รวดเร็ว (Kimball, 2011; Häfner, 2005).

1.2 วัตถุประสงค์

การสร้าง Multidimensional data model และแดชบอร์ดนำเสนอข้อมูลของฐานข้อมูล Chinook ผ่านมุมมองทางธุรกิจ มีวัตถุประสงค์ ดังนี้

1.2.1 สร้าง Multidimensional Data Model และออกแบบ Data Cube เพื่อจัดระเบียบข้อมูลในหลายมิติที่สามารถตอบสนองต่อคำถามทางธุรกิจที่หลากหลาย และเพื่อให้การวิเคราะห์ข้อมูลที่ซับซ้อนเป็นไปอย่างมีประสิทธิภาพ

1.2.2 สร้าง Dashboard เพื่อนำเสนอผลลัพธ์ เพื่อแสดงข้อมูลที่วิเคราะห์จาก Data Cube ในรูปแบบที่เข้าใจง่ายและรวดเร็ว ทำให้ผู้บริหารสามารถเข้าถึงและใช้ข้อมูลสำคัญในการตัดสินใจทางธุรกิจได้อย่างมีประสิทธิภาพ

1.3 ขอบเขตการศึกษา

1.3.1 ขอบเขตด้านข้อมูล

ข้อมูลที่ใช้ในการศึกษาครั้งนี้เป็นชุดข้อมูล Chinook ซึ่งเป็นฐานข้อมูลตัวอย่างที่มีข้อมูลเกี่ยวกับการขายเพลง รวมถึงข้อมูลเกี่ยวกับศิลปิน, อัลบั้ม, เพลง, ลูกค้า, และพนักงาน ข้อมูลเหล่านี้จะถูกนำมาใช้ในการสร้าง Multidimensional Data Model และ Data Cube รวมถึงการวิเคราะห์และนำเสนอผลลัพธ์ผ่าน Dashboard เพื่อให้ตอบสนองต่อคำถามทางธุรกิจที่สนใจ

1.3.2 ขอบเขตด้านความรู้และเนื้อหา

ขอบเขตด้านความรู้และเนื้อหาของการศึกษาครั้งนี้ครอบคลุมในเรื่องของการสร้างและการจัดการ Multidimensional Data Model และ Data Cube รวมถึงการออกแบบและการสร้าง Dashboard สำหรับการนำเสนอข้อมูล ขอบเขตนี้รวมถึงการศึกษาเทคนิคการวิเคราะห์ข้อมูลที่ซับซ้อน การสร้างรายงานและการสรุปผลจากข้อมูลที่มีมิติหลายด้าน และการนำเสนอผลลัพธ์ในรูปแบบที่เข้าใจง่ายและใช้ประโยชน์ได้ในบริบททางธุรกิจ

บทที่ 2

เอกสารที่เกี่ยวข้อง

2.1 คลังข้อมูล (Data Warehousing)

2.1.1 ความหมายของคลังข้อมูล

คลังข้อมูล (Data Warehouse) คือ ระบบการจัดเก็บข้อมูลที่ใช้สำหรับการรวบรวม จัดระเบียบ และจัดเก็บข้อมูลจากแหล่งข้อมูลหลายแห่ง โดยมุ่งเน้นให้ข้อมูลสามารถนำไปวิเคราะห์และใช้ในการตัดสินใจทางธุรกิจได้อย่างมีประสิทธิภาพ คลังข้อมูลถูกออกแบบมาเพื่อรองรับการวิเคราะห์ข้อมูลในลักษณะที่สามารถเข้าถึงได้ง่าย และทำให้ข้อมูลที่จัดเก็บมีความสอดคล้องกัน

2.1.2 คุณสมบัติของคลังข้อมูล

1. การรวมข้อมูลจากหลายแหล่ง (Data Integration)

คลังข้อมูลจะรวบรวมข้อมูลจากแหล่งข้อมูลที่หลากหลาย เช่น ฐานข้อมูลที่ทำงานในระหว่างวัน (Operational Databases), ระบบ CRM, ERP, และแหล่งข้อมูลภายนอกอื่น ๆ ข้อมูลที่ถูกดึงเข้ามาจะถูกรวมกันเพื่อให้มีความครอบคลุมและสามารถนำไปวิเคราะห์ได้อย่างมีประสิทธิภาพ

2. การปรับปรุงข้อมูล (Data Transformation)

ข้อมูลที่ถูกนำเข้ามาในคลังข้อมูลจะผ่านกระบวนการแปลง (Transformation) เพื่อให้ได้ข้อมูลที่มีคุณภาพสูง ซึ่งรวมถึงการลบข้อมูลซ้ำซ้อน การแก้ไขข้อผิดพลาด การจัดรูปแบบข้อมูลให้เหมาะสม และการสร้างการเชื่อมโยงระหว่างข้อมูลต่าง ๆ

3. การจัดเก็บข้อมูลแบบมิติ (Dimensional Storage)

คลังข้อมูลจะถูกออกแบบในรูปแบบที่เหมาะสมสำหรับการวิเคราะห์ โดยใช้โครงสร้างแบบมิติ เช่น star schema หรือ snowflake schema ซึ่งช่วยให้การสอบถามข้อมูลทำได้ง่ายและรวดเร็ว โดยสามารถวิเคราะห์ข้อมูลตามมิติต่าง ๆ เช่น เวลา สถานที่ และหมวดหมู่

4. การเก็บประวัติยาวนาน (Historical Data Storage)

คลังข้อมูลเก็บข้อมูลประวัติศาสตร์ที่มีความยาว ซึ่งช่วยให้ผู้ใช้สามารถทำการวิเคราะห์แนวโน้มและการเปลี่ยนแปลงในข้อมูลได้ รวมถึงการช่วยในการคาดการณ์ในอนาคต

5. การสนับสนุนการตัดสินใจ (Decision Support)

ข้อมูลในคลังข้อมูลถูกใช้เพื่อสนับสนุนการตัดสินใจทางธุรกิจ โดยสามารถสร้างรายงานและการวิเคราะห์ที่ช่วยให้ผู้บริหารสามารถตัดสินใจได้อย่างมีข้อมูลสนับสนุน

6. การอัปเดตแบบชุด (Batch Processing)

ข้อมูลในคลังข้อมูลมักจะถูกอัปเดตในรูปแบบชุด ซึ่งหมายความว่าจะมีการดึงข้อมูลจากแหล่งต้นทางในช่วงเวลาที่กำหนด (เช่น ทุกคืน) แทนที่จะอัปเดตแบบเรียลไทม์ การทำเช่นนี้ช่วยให้ประสิทธิภาพดีขึ้นและลดการโหลดในระบบต้นทาง

7. ความปลอดภัยและการเข้าถึง (Security and Access Control)

คลังข้อมูลมักจะมีมาตรการด้านความปลอดภัยในการเข้าถึงข้อมูล โดยมีการจัดการผู้ใช้และการกำหนดสิทธิ์ในการเข้าถึงข้อมูล เพื่อให้มั่นใจว่าข้อมูลที่สำคัญจะได้รับการป้องกันและเข้าถึงได้เฉพาะผู้ที่มีสิทธิ์

8. การเข้าถึงข้อมูลได้ง่าย (Easy Data Access)

ผู้ใช้สามารถเข้าถึงข้อมูลในคลังข้อมูลได้อย่างง่ายดาย โดยใช้เครื่องมือ BI และเครื่องมือวิเคราะห์ต่าง ๆ ที่ช่วยให้สามารถสอบถามและสร้างรายงานได้อย่างสะดวก

2.1.3 ข้อดี ข้อเสีย ของคลังข้อมูล

2.1.3.1. ข้อดีของคลังข้อมูล (Data Warehouse)

1. การตัดสินใจที่ดีขึ้น (Better Decision-Making) คลังข้อมูลช่วยให้ผู้ใช้งานสามารถเข้าถึงข้อมูลที่มีคุณภาพสูง ทำให้สามารถตัดสินใจได้อย่างแม่นยำและรวดเร็วขึ้น

2. รวมข้อมูลจากหลายแหล่ง (Data Integration) คลังข้อมูลสามารถรวบรวมข้อมูลจากหลายระบบหรือหลายแหล่งเข้าไว้ด้วยกัน ทำให้ได้ข้อมูลที่ครบถ้วนและมุมมองที่หลากหลาย

3. การเข้าถึงข้อมูลเชิงลึก (Historical Data Analysis) คลังข้อมูลจัดเก็บข้อมูลในระยะยาว ทำให้สามารถวิเคราะห์ข้อมูลในเชิงลึกและสามารถวิเคราะห์แนวโน้ม (Trend Analysis) ได้

4. การเพิ่มประสิทธิภาพการทำงาน (Improved Performance) การใช้คลังข้อมูลทำให้ระบบปฏิบัติการไม่ต้องรับภาระการสืบค้นที่ซับซ้อนในเวลาที่ผู้ใช้งานต้องการข้อมูล

5. ความถูกต้องและความสม่ำเสมอของข้อมูล (Data Consistency) การเก็บข้อมูลที่เป็นมาตรฐานและมีโครงสร้างเดียวกันช่วยลดความไม่ตรงกันของข้อมูลและทำให้เกิดความถูกต้องในข้อมูลที่ใช้

6. การสนับสนุนระบบการรายงานและวิเคราะห์ (Support for Reporting and Analytics) คลังข้อมูลช่วยให้การสร้างรายงานและการวิเคราะห์เป็นเรื่องง่ายและมีประสิทธิภาพมากขึ้น

2.1.3.2 ข้อเสียของคลังข้อมูล (Data Warehouse)

1. ค่าใช้จ่ายสูง (High Cost) การสร้างและดูแลรักษาคลังข้อมูลมีค่าใช้จ่ายสูง ทั้งในด้านฮาร์ดแวร์ ซอฟต์แวร์ และบุคลากรที่มีความเชี่ยวชาญ

2. ใช้เวลานานในการพัฒนา (Time-Consuming Development) การออกแบบและพัฒนาคลังข้อมูลใช้เวลานาน เนื่องจากต้องรวมข้อมูลจากหลายแหล่งและทำให้ข้อมูลมีความสมบูรณ์

3. ความซับซ้อนในการดูแลรักษา (Complex Maintenance) คลังข้อมูลต้องการการดูแลรักษาและอัพเดตข้อมูลอย่างสม่ำเสมอ เพื่อให้ข้อมูลใหม่ๆ ถูกจัดเก็บและพร้อมใช้งานเสมอ

4. การเปลี่ยนแปลงยาก (Difficulty in Adapting to Changes) หากโครงสร้างของข้อมูลต้นทางมีการเปลี่ยนแปลง จะทำให้คลังข้อมูลต้องมีการปรับปรุงตาม ซึ่งอาจทำให้การจัดการยุ่งยาก

5. ไม่เหมาะสำหรับข้อมูลเรียลไทม์ (Not Suitable for Real-Time Data) คลังข้อมูลส่วนใหญ่รองรับการประมวลผลข้อมูลที่มีการรวบรวมมาแล้ว และไม่เหมาะสำหรับการใช้งานข้อมูลเรียลไทม์

6. การรักษาความปลอดภัย (Security Concerns) เนื่องจากคลังข้อมูลมักมีข้อมูลสำคัญที่มีความละเอียดอ่อน การรักษาความปลอดภัยจึงเป็นสิ่งที่ต้องให้ความสำคัญ

2.2 กระบวนการ ETL (Extract, Transform, Load)

ระบบการดึงข้อมูล การแปลงข้อมูล และการโหลดข้อมูล (ETL) ของสภาพแวดล้อมคลังข้อมูลและธุรกิจอัจฉริยะ (DW/BI) ประกอบด้วยพื้นที่ทำงาน โครงสร้างข้อมูลที่ถูกสร้างขึ้น และชุดกระบวนการต่าง ๆ โดยระบบ ETL จะทำหน้าที่เชื่อมโยงระหว่างระบบข้อมูลต้นทางและพื้นที่นำเสนอใน DW/BI

2.2.1 การดึงข้อมูล (Extraction)

การดึงข้อมูลเป็นขั้นตอนแรกในการนำข้อมูลเข้าสู่สภาพแวดล้อมคลังข้อมูล ซึ่งหมายถึงการอ่านและทำความเข้าใจกับข้อมูลต้นทาง พร้อมทั้งคัดลอกข้อมูลที่จำเป็นเข้าสู่ระบบ ETL เพื่อการปรับแต่งในภายหลัง ในขั้นตอนนี้ ข้อมูลที่ดึงเข้ามาอยู่ในระบบ ETL แล้ว และพร้อมสำหรับการดำเนินการในกระบวนการถัดไป

2.2.2 การแปลงข้อมูล (Transformation)

เมื่อข้อมูลถูกดึงเข้ามาในระบบ ETL จะมีการปรับเปลี่ยนข้อมูลหลายอย่าง เช่น การทำความสะอาดข้อมูล (เช่น การแก้ไขการสะกดผิด การแก้ไขข้อขัดแย้งของโดเมน การจัดการกับข้อมูลที่หายไป หรือการแปลงข้อมูลให้อยู่ในรูปแบบมาตรฐาน) การรวมข้อมูลจากหลายแหล่ง และการลบข้อมูลซ้ำ

ระบบ ETL จะช่วยเพิ่มมูลค่าให้กับข้อมูลด้วยการทำงานเหล่านี้ โดยการปรับเปลี่ยนข้อมูลและเสริมข้อมูลให้มีคุณภาพสูงขึ้น ซึ่งกิจกรรมเหล่านี้สามารถสร้างข้อมูลเมตาดาต้าเพื่อวิเคราะห์ทางการแพทย์ในอนาคต โดยจะนำไปสู่การปรับปรุงคุณภาพข้อมูลในระบบต้นทาง

2.2.3 การโหลดข้อมูล (Load)

ขั้นตอนสุดท้ายของกระบวนการ ETL คือการจัดโครงสร้างข้อมูลในรูปแบบที่เหมาะสมและการโหลดข้อมูลเข้าสู่โมเดลมิติในพื้นที่นำเสนอ

ระบบ ETL จะส่งมอบตารางมิติและตารางข้อเท็จจริงในขั้นตอนนี้ โดยจะมีความสำคัญในการดำเนินการจัดการตารางมิติ เช่น การกำหนดกุญแจแทน การค้นหารหัสเพื่อนำเสนอคำอธิบายที่เหมาะสม การแยกหรือรวมคอลัมน์เพื่อนำเสนอค่าข้อมูลที่เหมาะสม หรือการรวมโครงสร้างตารางที่เป็นรูปแบบปกติสาม (3NF) เข้ากับมิติที่ไม่มีรูปแบบปกติ

2.3 เป้าหมายของคลังข้อมูลและระบบธุรกิจอัจฉริยะ (Goals of Data Warehousing and Business Intelligence)

ในการสร้างคลังข้อมูลและระบบธุรกิจอัจฉริยะ มีเป้าหมายสำคัญดังนี้

2.3.1 ระบบ DW/BI ต้องทำให้ข้อมูลเข้าถึงได้ง่าย

ระบบ DW/BI ต้องมีเนื้อหาที่สามารถทำความเข้าใจได้ง่ายและชัดเจนสำหรับผู้ใช้งานทางธุรกิจ ไม่เพียงแค่เข้าใจได้ง่ายสำหรับนักพัฒนาเท่านั้น โครงสร้างและการระบุชื่อต่างๆของข้อมูลควรใช้คำศัพท์ทางธุรกิจที่ทำให้ผู้ใช้งานทางธุรกิจสามารถเข้าใจความหมายของข้อมูลได้อย่างดี และผู้ใช้งานทางธุรกิจต้องสามารถแยกและรวมเนื้อหาที่ต้องการวิเคราะห์ธุรกิจได้อย่างครอบคลุม ระบบ DW/BI ต้องเรียบง่ายและใช้งานง่าย และต้องส่งคืนผลลัพธ์การค้นหาให้กับผู้ใช้งานโดยใช้เวลาไม่นาน

2.3.2 ระบบ DW/BI ต้องนำเสนอข้อมูลอย่างสม่ำเสมอ

ข้อมูลในระบบ DW/BI ต้องเชื่อถือได้ ข้อมูลต้องถูกรวบรวมอย่างระมัดระวังจากแหล่งข้อมูลที่หลากหลาย ทำการทำความสะอาด รับประกันคุณภาพ และปล่อยออกมาเมื่อข้อมูลนั้นเหมาะสมสำหรับการใช้งานของผู้ใช้ ความสม่ำเสมอยังหมายถึงการใช้ชื่อเรียกและคำจำกัดความที่เหมือนกันสำหรับเนื้อหาของระบบ DW/BI ข้ามแหล่งข้อมูล หากสองมาตรการประสิทธิภาพมีชื่อเดียวกันต้องหมายความเหมือนกัน หากมาตรการสองอย่างหมายความต่างกัน ต้องใช้ชื่อเรียกที่แตกต่างกัน

2.3.3 ระบบ DW/BI ต้องสามารถปรับตัวเข้ากับการเปลี่ยนแปลงได้

ความต้องการของผู้ใช้ สถานการณ์ทางธุรกิจ ข้อมูล และเทคโนโลยีทั้งหมดมีแนวโน้มที่จะเปลี่ยนแปลง ระบบ DW/BI ต้องถูกออกแบบมาให้สามารถจัดการกับการเปลี่ยนแปลงนี้ได้โดยไม่ทำให้ข้อมูลหรือแอปพลิเคชันที่มีอยู่เสียหาย ข้อมูลและแอปพลิเคชันที่มีอยู่ไม่ควรเปลี่ยนแปลงหรือถูกรบกวนเมื่อชุมชนธุรกิจมีคำถามใหม่หรือต้องการข้อมูลใหม่ถูกเพิ่มเข้าไปในคลังข้อมูล นอกจากนี้หากข้อมูลเชิงพรรณนาต้องมีการแก้ไข คุณต้องคำนึงถึงการเปลี่ยนแปลงนี้และทำให้การเปลี่ยนแปลงเหล่านี้โปร่งใสต่อผู้ใช้

2.3.4 ระบบ DW/BI ต้องนำเสนอข้อมูลในเวลาที่เหมาะสม

เมื่อระบบ DW/BI ถูกใช้มากขึ้นสำหรับการตัดสินใจในการดำเนินงาน ข้อมูลดิบอาจต้องถูกเปลี่ยนเป็นข้อมูลที่สามารถนำไปปฏิบัติได้ภายในไม่กี่ชั่วโมง นาที หรือแม้กระทั่งวินาที ทีม DW/BI และผู้ใช้ธุรกิจต้องมีความคาดหวังที่สมจริงสำหรับสิ่งที่หมายถึงการส่งมอบข้อมูลเมื่อไม่มีเวลาในการทำความสะอาดหรือยืนยันความถูกต้องของข้อมูล

2.3.5 ระบบ DW/BI ต้องเป็นป้อมปราการที่ปลอดภัยในการปกป้องข้อมูล

ขุมสมบัติข้อมูลขององค์กรมักจะถูกเก็บไว้ในคลังข้อมูล ขั้นต่ำที่สุด คลังข้อมูลอาจมีข้อมูลเกี่ยวกับสิ่งที่คุณขายให้กับใครในราคาเท่าไหร่ ข้อมูลที่อาจเป็นอันตรายในมือของผู้ที่ไม่เหมาะสม ระบบ DW/BI ต้องสามารถควบคุมการเข้าถึงข้อมูลที่เป็นความลับขององค์กรได้อย่างมีประสิทธิภาพ

2.3.6 ระบบ DW/BI ต้องเป็นฐานข้อมูลที่เชื่อถือได้เพื่อสนับสนุนการตัดสินใจที่ดีขึ้น

คลังข้อมูลต้องมีข้อมูลที่ถูกต้องในการสนับสนุนการตัดสินใจ ผลลัพธ์ที่สำคัญที่สุดจากระบบ DW/BI คือการตัดสินใจที่ทำโดยอิงจากหลักฐานเชิงวิเคราะห์ที่นำเสนอ การตัดสินใจเหล่านี้นำมาซึ่งผลกระทบทางธุรกิจและคุณค่าที่สามารถนำไปที่ระบบ DW/BI คำอธิบายที่เก่ากว่า ซึ่งหมายถึงระบบ DW/BI ยังคงเป็นคำอธิบายที่ดีที่สุดสำหรับสิ่งที่คุณออกแบบระบบสนับสนุนการตัดสินใจ

2.3.7 ชุมชนธุรกิจต้องยอมรับระบบ DW/BI เพื่อให้ถือว่าประสบความสำเร็จ

มันไม่สำคัญหรอกว่าคุณสร้างโซลูชันที่สวยงามโดยใช้ผลิตภัณฑ์และแพลตฟอร์มที่ดีที่สุด หากชุมชนธุรกิจไม่ยอมรับสภาพแวดล้อม DW/BI และไม่ใช้งานอย่างกระตือรือร้น คุณก็ไม่ผ่านการทดสอบการยอมรับนั้นแล้ว แม้ว่าในบางครั้งการใช้งาน DW/BI จะเป็นทางเลือก แต่ผู้ใช้งานในธุรกิจจะยอมรับระบบ DW/BI ถ้าหากมันเป็นแหล่งข้อมูลที่ “เรียบง่ายและรวดเร็ว” สำหรับข้อมูลที่สามารถนำไปปฏิบัติได้

แม้ว่าแต่ละข้อกำหนดในรายการทั้งหมดนี้จะมีความสำคัญ แต่สองข้อสุดท้ายเป็นข้อที่สำคัญที่สุด และน่าเสียดายที่มักถูกมองข้าม การที่จะประสบความสำเร็จในด้านการสร้างคลังข้อมูลและการวิเคราะห์ธุรกิจต้องการมากกว่าการเป็นสถาปนิก นักเทคนิค ผู้สร้างแบบจำลอง หรือผู้ดูแลฐานข้อมูลที่ยอดเยี่ยม ในโครงการ DW/BI คุณต้องมีความเชี่ยวชาญด้านเทคโนโลยีสารสนเทศ (IT) ขณะที่คุณก้าวไปสู่วิธีการที่ไม่คุ้นเคยกับผู้ใช้งานธุรกิจ คุณต้องปรับเปลี่ยนทักษะบางอย่างเพื่อให้สอดคล้องกับความต้องการที่ไม่เหมือนใครของ DW/BI อย่างชัดเจน คุณต้องนำเสนอทักษะที่หลากหลายเพื่อทำหน้าที่เป็น DBA/MBA ที่มีความหลากหลาย

2.4 การประมวลผลวิเคราะห์ออนไลน์ (OLAP)

2.4.1 แนะนำการสร้างแบบจำลองเชิงมิติ (Dimensional Modeling Introduction)

การสร้างแบบจำลองเชิงมิติเป็นเทคนิคที่มีมายาวนานในการทำให้ฐานข้อมูลมีความเรียบง่าย ตลอดหลายสิบปีที่ผ่านมา องค์กร IT ที่ปรึกษา และผู้ใช้ธุรกิจต่างมุ่งสู่โครงสร้างเชิงมิติที่ง่ายต่อการเข้าใจ เนื่องจากตอบสนองต่อความต้องการพื้นฐานของมนุษย์ที่ต้องการความเรียบง่าย ความเรียบง่ายเป็นสิ่งสำคัญเพราะทำให้ผู้ใช้สามารถเข้าใจข้อมูลได้ง่ายและทำให้ซอฟต์แวร์สามารถสืบค้นและส่งผลลัพธ์ได้อย่างรวดเร็วและมีประสิทธิภาพ

การสร้างแบบจำลองเชิงมิติ (Dimensional Modeling) เป็นเทคนิคที่ใช้กันอย่างแพร่หลายในการนำเสนอข้อมูลเชิงวิเคราะห์ในระบบ DW/BI เนื่องจากเน้นที่การทำให้ข้อมูลง่ายต่อการเข้าใจสำหรับผู้ใช้ธุรกิจและทำให้การสืบค้นข้อมูลมีประสิทธิภาพสูงสุด โดยการจัดโครงสร้างข้อมูลให้อยู่ในรูปแบบที่ง่ายต่อการใช้งาน เช่น การมองข้อมูลในรูปแบบลูกบาศก์ที่มีมิติสำคัญอย่างผลิตภัณฑ์ ตลาด และเวลา

การสร้างแบบจำลองเชิงมิติแตกต่างจากแบบจำลอง Third Normal Form (3NF) ซึ่งเน้นการกำจัดความซ้ำซ้อนของข้อมูล แม้แบบจำลอง 3NF จะมีประโยชน์ในการประมวลผลข้อมูลเชิงปฏิบัติการ แต่แบบจำลองนี้มีความซับซ้อนเกินไปสำหรับการสืบค้นเชิงวิเคราะห์ แบบจำลองเชิงมิติช่วยแก้ปัญหานี้โดยทำให้การสืบค้นข้อมูลเร็วและเข้าใจง่าย

โมเดลมิติที่ใช้ในระบบฐานข้อมูลเชิงสัมพันธ์มักถูกเรียกว่า Star Schema เนื่องจากโครงสร้างของมันคล้ายกับรูปดาว ในขณะที่โมเดลมิติที่ใช้ในสภาพแวดล้อมฐานข้อมูลมิติ (Multidimensional Database) จะเรียกว่า OLAP Cubes ทั้ง Star Schema และ OLAP Cube มีแนวคิดทางตรรกะเหมือนกันที่ใช้มิติ (Dimensions) ในการออกแบบ แต่การนำไปใช้งานจริงนั้นแตกต่างกัน

เมื่อข้อมูลถูกโหลดเข้าไปใน OLAP Cube ข้อมูลจะถูกจัดเก็บและจัดทำดัชนีโดยใช้รูปแบบและเทคนิคที่ออกแบบมาเพื่อรองรับข้อมูลมิติ ซึ่งจะมีการสร้างตารางสรุปหรือการคำนวณล่วงหน้าเพื่อเพิ่มประสิทธิภาพการประมวลผลการค้นหา ส่งผลให้ OLAP Cube มีประสิทธิภาพการค้นหาที่ดีกว่า เนื่องจากมีการคำนวณล่วงหน้าและใช้กลยุทธ์การจัดทำดัชนีที่ซับซ้อน

ผู้ใช้งานสามารถ "drill down" หรือ "drill up" โดยการเพิ่มหรือลบมิติจากการวิเคราะห์ได้อย่างรวดเร็วโดยไม่ต้องส่งคำสั่งค้นหาใหม่ นอกจากนี้ OLAP Cubes ยังมีฟังก์ชันการวิเคราะห์ที่ซับซ้อนกว่า SQL แต่ข้อเสียคือ การโหลดข้อมูลเข้า OLAP Cube อาจใช้เวลานาน โดยเฉพาะเมื่อข้อมูลมีขนาดใหญ่

2.4.2 การสร้าง Multidimensional data model

การสร้าง Multidimensional Data Model หรือ โมเดลข้อมูลเชิงมิติ คือการออกแบบและจัดระเบียบข้อมูลในลักษณะที่ช่วยให้การวิเคราะห์และการรายงานข้อมูลเป็นไปได้อย่างมีประสิทธิภาพ โดยเฉพาะในบริบทของการวิเคราะห์ข้อมูลขนาดใหญ่และระบบ OLAP (Online Analytical Processing) โมเดลเชิงมิติมักใช้ในการสร้าง Data Warehouse หรือ Data Mart เพื่อการวิเคราะห์และรายงานเชิงลึก

2.4.2.1 แนวคิดหลักของโมเดลข้อมูลเชิงมิติ

1. มิติ (Dimensions)

มิติหมายถึงลักษณะหรือเกณฑ์ที่ใช้ในการจัดกลุ่มข้อมูล เช่น เวลา (Year, Month, Day), สถานที่ (City, State, Country), และผลิตภัณฑ์ (Product Category, Product Name) เป็นต้น

ตารางมิติ (Dimension Table) คือ ตารางที่เก็บข้อมูลเกี่ยวกับมิติ เช่น ตารางที่เก็บข้อมูลเกี่ยวกับลูกค้า ผลิตภัณฑ์ หรือเวลา ข้อมูลในตารางมิติมักจะเป็นข้อมูลที่ใช้ในการอธิบายหรือจัดกลุ่มข้อเท็จจริง

2. ข้อเท็จจริง (Facts)

ข้อเท็จจริงคือข้อมูลที่สามารถวัดได้และใช้ในการวิเคราะห์ เช่น ยอดขาย (Sales Revenue), ปริมาณขาย (Quantity Sold) เป็นต้น ข้อมูลข้อเท็จจริงมักถูกจัดเก็บในรูปแบบของ Fact Tables ซึ่งจะมีค่าตัวเลขที่สามารถนำมาวิเคราะห์

ตารางข้อเท็จจริง (Fact Table) คือ ตารางที่เก็บข้อมูลข้อเท็จจริงพร้อมกับคีย์ที่เชื่อมโยงไปยังมิติ ข้อมูลในตารางข้อเท็จจริงมักจะเป็นข้อมูลที่ต้องการวิเคราะห์ เช่น ยอดขายรายเดือน หรือกำไรประจำไตรมาส

3. ดาว (Star Schema) และ สโนว์เฟล (Snowflake Schema)

ดาวและสโนว์เฟลเป็นรูปแบบของการจัดระเบียบข้อมูลในโมเดลเชิงมิติ โดยมีลักษณะของดาวและสโนว์เฟล ดังนี้

Star Schema ใช้ตารางข้อเท็จจริงเชื่อมต่อกับตารางมิติผ่านคีย์หลัก ทำให้โครงสร้างมีลักษณะคล้ายดาว

Snowflake Schema ขยายรูปแบบดาวโดยการจัดระเบียบตารางมิติให้มีลักษณะเป็นลำดับชั้น ทำให้โครงสร้างมีลักษณะคล้ายหิมะ

2.5 การสร้าง Web Application ด้วย Streamlit

2.5.1 ความหมายของเว็บแอปพลิเคชัน

เว็บแอปพลิเคชัน (Web Application) คือซอฟต์แวร์ที่สามารถทำงานได้ผ่านเว็บเบราว์เซอร์ ผู้ใช้สามารถโต้ตอบกับแอปพลิเคชันผ่านอินเทอร์เน็ตโดยไม่จำเป็นต้องติดตั้งโปรแกรมลงในเครื่องคอมพิวเตอร์ ซึ่งเว็บแอปพลิเคชันจะแตกต่างจากเว็บไซต์ทั่วไปตรงที่สามารถมีฟังก์ชันโต้ตอบ (Interactive) และทำงานได้อย่างซับซ้อน เช่น แอปพลิเคชันสำหรับการจัดการข้อมูล การวิเคราะห์ข้อมูล การชำระเงินออนไลน์ หรือการรายงานผล

2.5.2 Streamlit

Streamlit เป็นโอเพนซอร์สเฟรมเวิร์กที่ออกแบบมาเพื่อให้นักพัฒนาสามารถสร้างเว็บแอปพลิเคชันสำหรับการวิเคราะห์ข้อมูลและแสดงผลข้อมูลได้อย่างรวดเร็ว โดยใช้เพียงภาษา Python นักพัฒนาสามารถสร้างอินเทอร์เฟซที่ตอบสนองและมีความยืดหยุ่นโดยไม่จำเป็นต้องมีความรู้ลึกซึ้งเกี่ยวกับ HTML, CSS หรือ JavaScript ซึ่งแตกต่างจากเฟรมเวิร์กอื่นๆ ที่ต้องใช้หลายภาษาในการพัฒนาเว็บแอป

2.5.2.1 ข้อดีของ Streamlit

1. ใช้งานง่าย: เขียนโค้ด Python แล้วแปลงเป็นแอปพลิเคชันได้ทันที

2. ตอบสนองทันที: การเปลี่ยนแปลงในโค้ดจะแสดงผลโดยตรงในแอป

3. โต้ตอบได้: สนับสนุนการรับค่าจากผู้ใช้ เช่น ฟอร์มการกรอกข้อมูล, การเลือกค่าจากลิสต์, หรือการเลื่อนสไลเดอร์

4. ไม่ต้องการความรู้ด้านเว็บโปรแกรมมิ่ง: นักพัฒนาเพียงแค่ใช้ภาษา Python ในการสร้างแอปพลิเคชัน

5. รองรับไลบรารีต่าง ๆ: Streamlit สามารถทำงานร่วมกับไลบรารีสำหรับการวิเคราะห์ข้อมูล เช่น Pandas, NumPy, Matplotlib, Plotly และอื่น ๆ

6. การพัฒนาแอปพลิเคชันอย่างรวดเร็ว: การเขียนโค้ดด้วย Streamlit ช่วยให้การสร้างเว็บแอปพลิเคชันใช้เวลาเพียงไม่กี่นาที

2.5.3 หลักการพื้นฐานของ Streamlit

Streamlit เป็น Python framework ที่ถูกออกแบบมาเพื่อช่วยนักพัฒนาสร้างเว็บแอปพลิเคชันที่มีการโต้ตอบกับผู้ใช้อย่างง่ายดาย โดยมุ่งเน้นไปที่การพัฒนาแอปพลิเคชันสำหรับงานวิทยาศาสตร์ข้อมูล (Data Science) และการเรียนรู้ของเครื่อง (Machine Learning) เนื่องจากการสร้างแอปพลิเคชันบน Streamlit สามารถแสดงผลข้อมูล กราฟต่าง ๆ และการวิเคราะห์ข้อมูลในแบบเรียลไทม์ได้

2.5.4 การทำงานของ Streamlit

Streamlit ทำงานโดยการ "แปลง" สคริปต์ Python ให้กลายเป็นแอปพลิเคชันเว็บแบบโต้ตอบได้โดยอัตโนมัติ ผ่านชุดของฟังก์ชันที่ช่วยในการแสดงผลและโต้ตอบกับผู้ใช้ เช่น การแสดงผลข้อมูล การสร้างกราฟ การรับข้อมูลจากผู้ใช้ เป็นต้น ทุกครั้งที่มีการแก้ไขหรืออัปเดตโค้ดแอปพลิเคชันจะถูกรีเฟรชอัตโนมัติและทำงานใหม่อีกครั้ง

Streamlit สามารถรองรับการทำงานแบบ Stateful คือ แอปพลิเคชันจะจำค่าต่าง ๆ ที่ผู้ใช้กรอกหรือเลือกได้ ทำให้สามารถทำงานในลักษณะโต้ตอบได้อย่างสมบูรณ์

2.6 รายละเอียดของข้อมูลในฐานข้อมูล Chinook

ฐานข้อมูล Chinook มีตาราง ดังนี้

2.6.1 ตาราง employees: เก็บข้อมูลพนักงาน เช่น รหัสพนักงาน, นามสกุล, ชื่อ เป็นต้น นอกจากนี้ยังมีฟิลด์ที่ชื่อว่า ReportsTo ซึ่งใช้ระบุว่าพนักงานคนใดรายงานถึงพนักงานคนใด

2.6.2 ตาราง customers: เก็บข้อมูลลูกค้า

2.6.3 ตาราง invoices และ invoice_items: ตาราง invoices เก็บข้อมูลส่วนหัวของใบแจ้งหนี้ และตาราง invoice_items เก็บข้อมูลรายการในใบแจ้งหนี้

2.6.4 ตาราง artists: เก็บข้อมูลศิลปิน ซึ่งเป็นตารางที่ประกอบด้วยรหัสศิลปินและชื่อ

2.6.5 ตาราง albums: เก็บข้อมูลเกี่ยวกับอัลบั้มเพลง โดยแต่ละอัลบั้มจะเป็นของศิลปินหนึ่งคน แต่ศิลปินหนึ่งคนอาจมีหลายอัลบั้ม

2.6.6 ตาราง media_types: เก็บข้อมูลประเภทสื่อ เช่น ไฟล์เสียง MPEG และ AAC

2.6.7 ตาราง genres: เก็บข้อมูลประเภทของเพลง เช่น ร็อก, แจ๊ส, เมทัล เป็นต้น

2.6.8 ตาราง tracks: เก็บข้อมูลของเพลง โดยแต่ละเพลงจะเป็นของอัลบั้มหนึ่งอัลบั้ม

2.6.9 ตาราง playlists และ playlist_track: ตาราง playlists เก็บข้อมูลเกี่ยวกับเพลย์ลิสต์ โดยแต่ละเพลย์ลิสต์ประกอบด้วยรายการเพลง และแต่ละเพลงอาจอยู่ในเพลย์ลิสต์หลายรายการ ความสัมพันธ์ระหว่างตาราง playlists และ tracks เป็นแบบหลายต่อหลาย (many-to-many) ซึ่งตาราง playlist_track ใช้ในการสะท้อนความสัมพันธ์นี้

บทที่ 3

วิธีการดำเนินการดำเนินโครงงาน

3.1 มุมมองธุรกิจที่สนใจ

ในการศึกษาครั้งนี้เราได้มุ่งเน้นการวิเคราะห์ข้อมูลยอดขายเพลงผ่านมุมมองธุรกิจที่สำคัญ เพื่อช่วยในการทำความเข้าใจภาพรวมของตลาดเพลง รวมถึงการประเมินประสิทธิภาพของศิลปิน แนวเพลง พนักงาน และภูมิภาคต่าง ๆ ที่มีส่วนในการสร้างรายได้ โดยมุมมองธุรกิจที่สนใจในการดำเนินโครงงานนี้ประกอบไปด้วย 10 มุมมองหลัก ดังนี้

3.1.1 ยอดขายรวมของเพลงทั้งหมด

มุมมองนี้จะวิเคราะห์ยอดขายรวมของเพลงทั้งหมดในช่วงเวลาที่ผ่านมา เพื่อให้เห็นภาพรวมของตลาดเพลง และช่วยในการวัดขนาดและศักยภาพของธุรกิจเพลงในปัจจุบัน

3.1.2 ยอดขายรวมของเพลงในแต่ละช่วงเวลา

วิเคราะห์ยอดขายในแต่ละปี เพื่อทำความเข้าใจแนวโน้มการเติบโตหรือถดถอยของยอดขายเพลงในช่วงเวลาต่าง ๆ ซึ่งช่วยให้สามารถตรวจสอบการเปลี่ยนแปลงของตลาดเพลงได้

3.1.3 ยอดขายตามประเภทของเพลง

ศึกษายอดขายของเพลงตามแนวเพลง เพื่อระบุแนวเพลงที่ได้รับความนิยมสูงและมีการสร้างรายได้มากที่สุด

3.1.4 Top 5 ศิลปินที่มียอดขายสูงสุดในแต่ละปี

มุมมองนี้มุ่งเน้นการระบุศิลปินที่ประสบความสำเร็จสูงสุดในแต่ละปี ซึ่งข้อมูลนี้สามารถใช้ในการวางแผนการสนับสนุนหรือส่งเสริมศิลปินได้

3.1.5 Top 5 ประเทศที่มียอดซื้อสูงสุดในปีล่าสุด

การวิเคราะห์ประเทศที่มียอดซื้อเพลงสูงสุดในปีล่าสุด จะช่วยให้เห็นถึงภูมิภาคที่มีศักยภาพในการขยายตลาดเพิ่มเติม หรือเน้นการตลาดในพื้นที่ที่มีการเติบโต

3.1.6 ยอดขายของพนักงานในแต่ละปี

มุมมองนี้จะช่วยให้ผู้บริหารสามารถติดตามประสิทธิภาพการทำงานของพนักงานในการสร้างยอดขายในแต่ละปี ซึ่งเป็นข้อมูลที่สำคัญในการประเมินและพัฒนาทรัพยากรบุคคล

3.1.7 ยอดขายตามภูมิภาค

วิเคราะห์ยอดขายตามภูมิภาคเพื่อประเมินความสำเร็จของธุรกิจในพื้นที่ต่าง ๆ ซึ่งสามารถนำข้อมูลนี้ไปใช้ในการวางกลยุทธ์ทางการตลาดในอนาคต

3.1.8 แนวโน้มยอดขายเพลงในแต่ละแนวเพลง

มุมมองนี้จะช่วยในการวิเคราะห์แนวโน้มการเปลี่ยนแปลงของความนิยมในแนวเพลงต่าง ๆ โดยสามารถใช้ข้อมูลนี้ในการคาดการณ์แนวโน้มตลาดในอนาคตได้

3.1.9 พนักงานที่มียอดขายสูงสุดในแต่ละปี

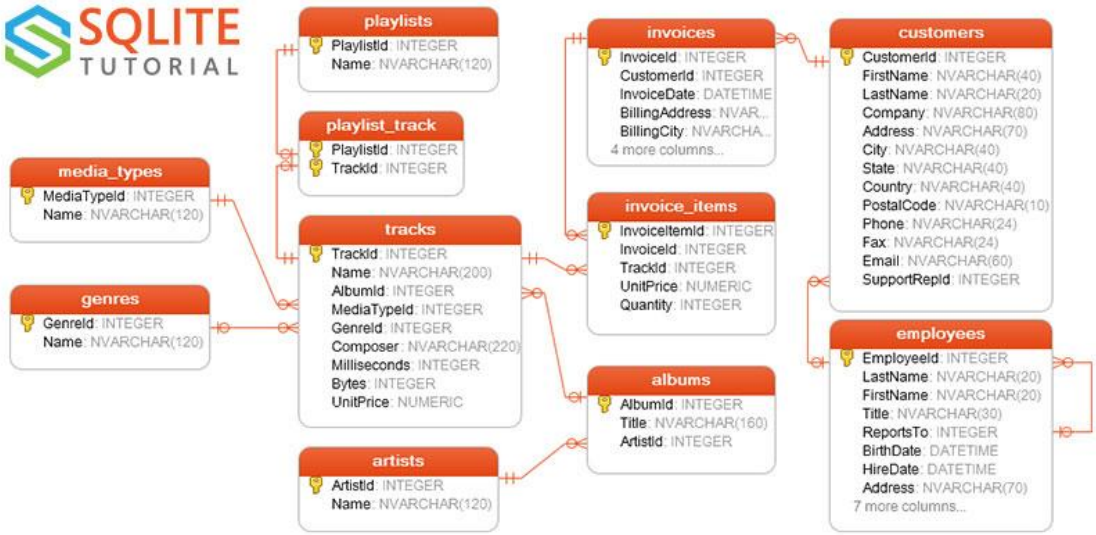
มุ่งเน้นการระบุพนักงานที่มีผลงานโดดเด่นที่สุดในแต่ละปี เพื่อเป็นข้อมูลในการให้รางวัลหรือพัฒนาศักยภาพของพนักงานต่อไป

3.1.10 พนักงานที่มียอดขายสูงสุดโดยรวม

การตรวจสอบพนักงานที่มียอดขายสูงสุดโดยรวมจะช่วยในการระบุพนักงานที่มีผลงานสม่ำเสมอและเป็นกำลังสำคัญของธุรกิจ

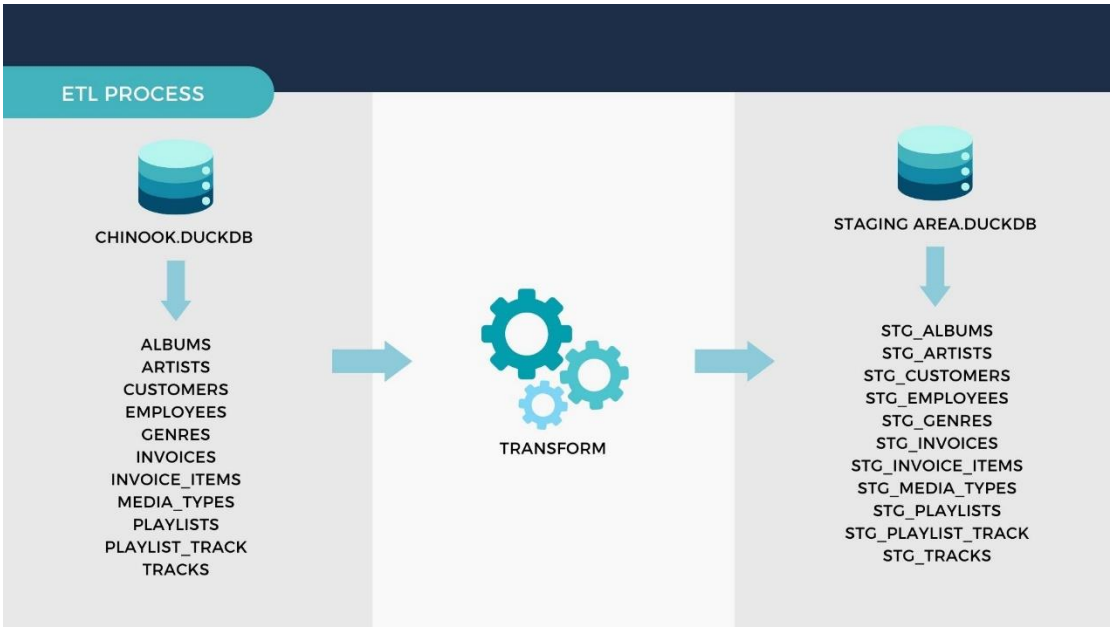
มุมมองเหล่านี้เป็นจุดเริ่มต้นสำคัญที่ช่วยให้สามารถวิเคราะห์ธุรกิจเพลงได้รอบด้านและนำข้อมูลที่ได้มาใช้ในการตัดสินใจที่มีประสิทธิภาพ

3.2 Entity-Relationship (ER) Diagram of Operational Database



ภาพที่ 3.1 ER-Diagram of Operational Database

3.3 ETL Process



ภาพที่ 3.3 ETL Process

3.3.1 นำเข้า Library ที่จำเป็น

```
import duckdb as dd
import polars as pl
import pandas as pd
import dlt
import sqlite3
from datetime import datetime
```

3.3.2 เชื่อมต่อกับฐานข้อมูล Chinook Database และแสดงตารางทั้งหมดใน Chinook Database

```
#path
db_path=r'C:\DataWarehouse_Project\chinook.db'
try:
    conn = sqlite3.connect(db_path)
    print("Connection successful")
except Exception as e:
    print(f"Error: {e}")

#show all Table
def show_tables(connection):
  """Shows all tables in the connected database."""
  cursor = connection.cursor()
  cursor.execute("SELECT name FROM sqlite_master WHERE type='table';")
  tables = cursor.fetchall()
  for table in tables:
      print(table[0])

show_tables(conn)
```

3.3.3 extraction data and Close Connection

```
#Extract
albums = pl.read_database(
   query= "SELECT * FROM albums",
   connection=conn )
sqlite_sequence = pl.read_database(
   query= "SELECT * FROM sqlite_sequence",
   connection=conn )
sqlite_sequence
artists = pl.read_database(
   query= "SELECT * FROM artists",
   connection=conn )
artists
customers = pl.read_database(
   query= "SELECT * FROM customers",
   connection=conn )
customers
employees = pl.read_database(
   query= "SELECT * FROM employees",
   connection=conn )
employees
genres = pl.read_database(
   query= "SELECT * FROM genres",
   connection=conn )
genres
invoice_items = pl.read_database(
   query= "SELECT * FROM invoice_items",
   connection=conn )
invoice_items
invoices = pl.read_database(
   query= "SELECT * FROM invoices",
   connection=conn )
invoices
```

```
media_types = pl.read_database(
   query= "SELECT * FROM media_types",
   connection=conn )
media_types
playlist_track = pl.read_database(
   query= "SELECT * FROM playlist_track",
   connection=conn )
playlist_track
playlists = pl.read_database(
   query= "SELECT * FROM playlists",
   connection=conn )
playlists
tracks = pl.read_database(
   query= "SELECT * FROM tracks",
   connection=conn )
tracks
sqlite_stat1 = pl.read_database(
    query= "SELECT * FROM sqlite_stat1",
    connection=conn )
sqlite_stat1
conn.close()
```

3.3.4 Create Function

```
#create Function
def rename_col(df, colname_dict):
        for old_name, new_name in colname_dict.items():
            df = df.rename({old_name: new_name})
        return df

def add_timestamp(df, colname):
        current_timestamp = datetime.now()
        df_n = df.with_columns(pl.lit(current_timestamp).alias(colname))
        return df_n
```

```
def unique(df, colname):
        return df.unique(subset=[colname])

def sort(df, colname):
        return df.sort(by=colname)

def rename_col(df, colname_dict):
        for old_name, new_name in colname_dict.items():
            df = df.rename({old_name: new_name})
        return df

def exclude(df, colname):
        return df.select(pl.col('*').exclude(colname))

def convert_str_to_datetime(df, column_name,
datetime_format='%Y-%m-%d %H:%M:%S'):
    return df.with_columns(
        pl.col(column_name).str.strptime(pl.Datetime,
format=datetime_format).alias('Date_time'))

def group_by_and_sum(df: pl.DataFrame, groupby_col: str, sum_col: str) ->
pl.DataFrame:
    return
df.group_by(groupby_col).agg(pl.col(sum_col).sum().alias(f"{sum_col}_sum"))

def group_by_and_count(df: pl.DataFrame, groupby_col: str) -> pl.DataFrame:
    return df.group_by(groupby_col).agg(pl.count().alias('count'))

def split_month_year(df, datetime_col):
    # Ensure the datetime_col is in datetime format
    df = df.with_columns(
        pl.col(datetime_col).cast(pl.Datetime).alias(datetime_col))
        # Add new columns for month and year
    return df.with_columns([
        pl.col(datetime_col).dt.month().alias(f"{datetime_col}_Month"),
        pl.col(datetime_col).dt.year().alias(f"{datetime_col}_Year")])
```

3.3.5 Transform Data

```
#Tranform albums
albums
Al_f=(albums.pipe(rename_col,{"AlbumId":"Al_Id"})
      .pipe(rename_col,{"Title":"Al_Ti"})
      .pipe(rename_col,{"ArtistId":"Ar_Id"})
      .pipe(add_timestamp,'Time_Stamp')
).to_pandas()
Al_f

#Tranform artists
artists
Ar_f=(
    artists.pipe(rename_col,{"ArtistId":"Ar_Id"})
    .pipe(rename_col,{"Name":"Ar_Name"})
    .pipe(add_timestamp,'Time_Stamp')
).to_pandas()
Ar_f

#Tranform Customers
customers.columns
Cus_f = (
    customers.pipe(rename_col, {"CustomerId": "Cus_Id"})  # เปลี่ยนชื่อคอลัมน์
      .pipe(rename_col, {"LastName": "Cus_L"})
      .pipe(rename_col, {"FirstName": "Cus_F"})
      .pipe(rename_col, {"Company": "Cus_Com"})
      .pipe(rename_col, {"Address": "Cus_Ad"})
      .pipe(rename_col, {"City": "Cus_City"})
      .pipe(rename_col, {"State": "Cus_State"})
      .pipe(rename_col, {"Country": "Cus_Contry"})
      .pipe(rename_col, {"PostalCode": "Cus_Pos"})
      .pipe(rename_col, {"Phone": "Cus_Phone"})
      .pipe(rename_col, {"Fax": "Cus_Fax"})
      .pipe(rename_col, {"Email": "Cus_Email"})
      .pipe(rename_col, {"SupportRepId": "Em_Id"})
```

```
      .pipe(add_timestamp,'Time_Stamp')
).to_pandas()
Cus_f

# Transformation process using .pipe
Em_f = (
    employees.pipe(rename_col, {"EmployeeId": "Em_Id"})  # เปลี่ยนชื่อคอลัมน์
      .pipe(rename_col, {"LastName": "Em_L"})
      .pipe(rename_col, {"FirstName": "Em_F"})
      .pipe(rename_col, {"Title": "Em_Ti"})
      .pipe(rename_col, {"BirthDate": "Em_Birth"})
      .pipe(rename_col, {"HireDate": "Em_Hire"})
      .pipe(rename_col, {"Address": "Em_Ad"})
      .pipe(rename_col, {"City": "Em_City"})
      .pipe(rename_col, {"State": "Em_State"})
      .pipe(rename_col, {"Country": "Em_Contry"})
      .pipe(rename_col, {"PostalCode": "Em_Pos"})
      .pipe(rename_col, {"Phone": "Em_Phone"})
      .pipe(rename_col, {"Fax": "Em_Fax"})
      .pipe(rename_col, {"Email": "Em_Email"})
      .pipe(add_timestamp,'Time_Stamp')
).to_pandas()
Em_f

#Tranform genres
genres
Ge_f=(
      genres.pipe(rename_col,{"GenreId":"Ge_Id"})
      .pipe(rename_col,{"Name":"Ge_Name"})
      .pipe(add_timestamp,'Time_Stamp')
).to_pandas()
Ge_f
```

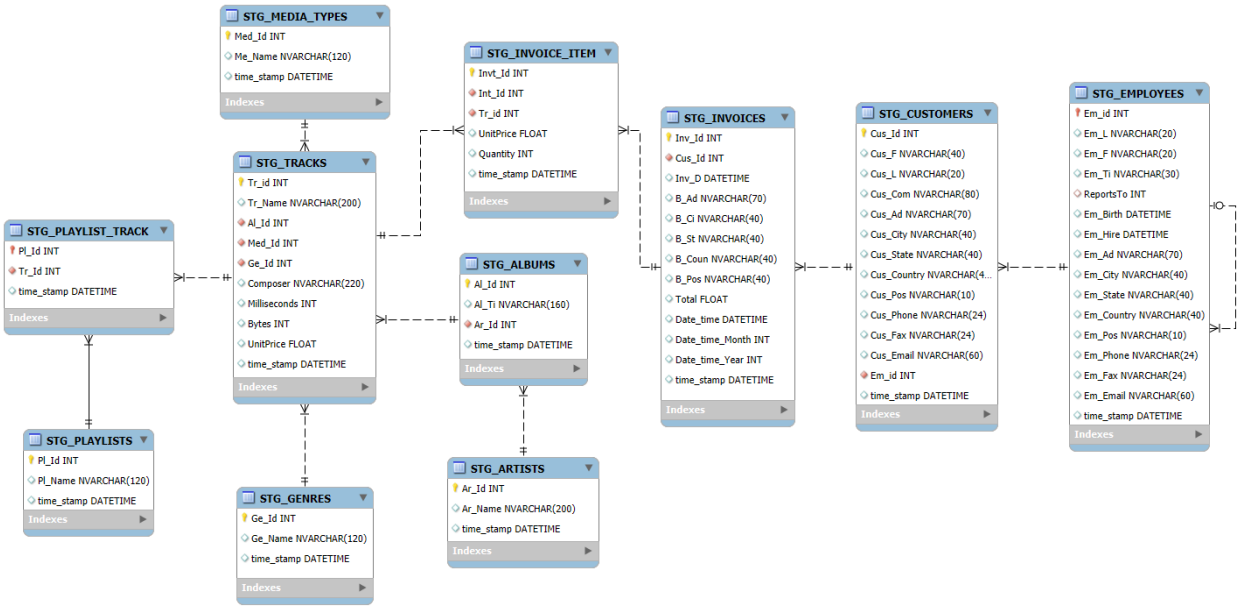
```
#Tranform invoices
invoices.columns
Inv_f=(
    invoices.pipe(rename_col, {"InvoiceId": "Inv_Id"})
    .pipe(rename_col, {"CustomerId": "Cus_Id"})
    .pipe(rename_col, {"InvoiceDate": "Inv_D"})
    .pipe(rename_col, {"BillingAddress": "B_Ad"})
    .pipe(rename_col, {"BillingCity": "B_Ci"})
    .pipe(rename_col, {"BillingState": "B_St"})
    .pipe(rename_col, {"BillingCountry": "B_Coun"})
    .pipe(rename_col, {"BillingPostalCode": "B_Pos"})
    .pipe(convert_str_to_datetime, 'Inv_D')  # Convert to datetime format
    .pipe(split_month_year,'Date_time')          # Split into month and year
    .pipe(add_timestamp,'Time_Stamp')
   ).to_pandas()
Inv_f

#Tranform invoices_items
invoice_items
Invt_f=(
      invoice_items.pipe(rename_col,{"InvoiceLineId":"Invt_Id"})
      .pipe(rename_col,{"InvoiceId":"Inv_Id"})
      .pipe(rename_col,{"TrackId":"Tr_Id"})
      .pipe(add_timestamp,'Time_Stamp')
).to_pandas()
Invt_f

#Tranform media_types
media_types
Med_f=(
      media_types.pipe(rename_col,{"MediaTypeId":"Med_Id"})
      .pipe(rename_col,{"Name":"Me_Name"})
      .pipe(add_timestamp,'Time_Stamp')
).to_pandas()
Med_f
```

```
#Tranform playlists
playlists
Pl_f=(
      playlists.pipe(rename_col,{"PlaylistId":"Pl_Id"})
      .pipe(rename_col,{"Name":"Pl_Name"})
      .pipe(add_timestamp,'Time_Stamp')
).to_pandas()
Pl_f

#Tranform playlist_tracks
playlist_track
PlT_f=(
      playlist_track.pipe(rename_col,{"PlaylistId":"Pl_Id"})
      .pipe(rename_col,{"TrackId":"Tr_Id"})
      .pipe(add_timestamp,'Time_Stamp')
).to_pandas()
PlT_f

#Tranform Track
tracks
tracks.columns
Tr_f=(
      tracks.pipe(rename_col,{"TrackId":"Tr_Id"})
      .pipe(rename_col,{"Name":"Tr_Name"})
      .pipe(rename_col,{"AlbumId":"Al_Id"})
      .pipe(rename_col,{"MediaTypeId":"Med_Id"})
      .pipe(rename_col,{"GenreId":"Ge_Id"})
      .pipe(add_timestamp,'Time_Stamp')
).to_pandas()
Tr_f
```

3.3.6 Load Data

```
pipeline=dlt.pipeline(
    pipeline_name="Operational",destination='duckdb',
    dataset_name="Project_stg"
)

pipeline.run(Al_f,table_name="stg_Albums",write_disposition='append')
pipeline.run(Ar_f,table_name="stg_Artists",write_disposition='append')
pipeline.run(Cus_f,table_name="stg_customers",write_disposition='append')
pipeline.run(Em_f,table_name="stg_employee",write_disposition='append')
pipeline.run(Ge_f,table_name="stg_Genres",write_disposition='append')
pipeline.run(Inv_f,table_name="stg_Invoices",write_disposition='append')
pipeline.run(Invt_f,table_name="stg_Invoices_items",write_disposition='append')
pipeline.run(Med_f,table_name="stg_Media_Types",write_disposition='append')
pipeline.run(Pl_f,table_name="stg_Playlist",write_disposition='append')
pipeline.run(PlT_f,table_name="stg_Playlist_Tracks",write_disposition='append')
pipeline.run(Tr_f,table_name="stg_Tracks",write_disposition='append')
```

3.4 Entity-Relationship (ER) Diagram of Staging area Database



ภาพที่ 3.2 ER-Diagram of Staging area Database

3.5 Data Dictionary of Entity-Relationship (ER) Diagram of Staging area Database

3.5.1 Table : STG_EMPLOYEES

Description : ตารางจัดเก็บข้อมูลเกี่ยวกับพนักงาน

ชื่อฟิลด์	ชนิดข้อมูล	ความยาว	คีย์	รายละเอียด	ตัวอย่างข้อมูล
Em_id	Int		PK	รหัสพนักงาน	1,2,3,4
Em_L	Nvarchar	20		นามสกุลพนักงาน	Adams
Em_F	Nvarchar	20		ชื่อจริงพนักงาน	Andrew
Em_Ti	Nvarchar	30		ชื่อตำแหน่งพนักงาน	General Manager
ReportsTo	Int		FK	รหัสหัวหน้าของพนักงาน	1,2,3,4
Em_Birth	Datetime			วันเกิดพนักงาน	1962-02-18 00:00:00
Em_Hire	Datetime			วันที่จ้างพนักงาน	2002-08-14 00:00:00
Em_Ad	Nvarchar	70		ที่อยู่ของพนักงาน	11120 Jasper Ave NW
Em_City	Nvarchar	40		เมืองของพนักงาน	Edmonton
Em_State	Nvarchar	40		รัฐของพนักงาน	AB
Em_Country	Nvarchar	40		ประเทศของพนักงาน	Canada

Em_Pos	Nvarchar	10		รหัสไปรษณีย์ของพนักงาน	T5K2N1
Em_Phone	Nvarchar	24		เบอร์โทรศัพท์ของพนักงาน	+1(780)428-9482
Em_Fax	Nvarchar	24		แฟกซ์ของพนักงาน	+1(780)428-3457
Em_Email	Nvarchar	60		อีเมลของพนักงาน	andrew@chinookcorp.com
time_stamp	datetime			เวลาที่ข้อมูลเข้า	22.30 01/01/2023

3.5.2 Table : STG_CUSTOMERS

Description : ตารางจัดเก็บข้อมูลเกี่ยวกับลูกค้า

ชื่อฟิลด์	ชนิดข้อมูล	ความยาว	คีย์	รายละเอียด	ตัวอย่างข้อมูล
Cus_id	Int		PK	รหัสลูกค้า	1,2,3,4
Cus_F	Nvarchar	40		ชื่อจริงของลูกค้า	Adams
Cus_L	Nvarchar	20		นามสกุลของลูกค้า	Andrew
Cus_Com	Nvarchar	80		บริษัทของลูกค้า	JetBrains s.r.o.
Cus_Ad	Nvarchar	70		ที่อยู่ของลูกค้า	Theodor-Heuss-Straße 34

Cus_City	Nvarchar	40		เมืองของลูกค้า	Edmonton
Cus_State	Nvarchar	40		รัฐของลูกค้า	AB
Cus_Country	Nvarchar	40		ประเทศของลูกค้า	Canada
Cus_Pos	Nvarchar	10		รหัสไปรษณีย์ของลูกค้า	T5K2N1
Cus_Phone	Nvarchar	24		เบอร์โทรศัพท์ของลูกค้า	+1(780)428-9482
Cus_Fax	Nvarchar	24		แฟกซ์ของลูกค้า	+1(780)428-3457
Cus_Email	Nvarchar	60		อีกเมลของลูกค้า	luisg@embraer.com.br
Em_id	Int		FK	รหัสของพนักงานที่ทำการดูแล	1,2,3,4
time_stamp	datetime			เวลาที่ข้อมูลเข้า	22.30 01/01/2023

3.5.3 Table : STG_INVOICES

Description : ตารางเก็บขอมูลเกี่ยวกับใบเสร็จ

ชื่อฟิลด์	ชนิดข้อมูล	ความยาว	คีย์	รายละเอียด	ตัวอย่างข้อมูล
Inv_Id	Int		PK	รหัสใบเสร็จ	1,2,3,4
Cus_Id	Int		FK	รหัสลูกค้า	1,2,3,4
Inv_D	Datetime			วันที่ออกในเสร็จ	2024-08-10

B_Ad	Nvarchar	70		ที่อยู่ในใบเสร็จ	Theodor-Heuss-Straße 34
B_Ci	Nvarchar	40		เมืองในที่อยู่ใบเสร็จ	Edmonton
B_St	Nvarchar	40		รัฐในที่อยู่ใบเสร็จ	AB
B_Coun	Nvarchar	40		ประเทศในที่อยู่ใบเสร็จ	Canada
B_Pos	Nvarchar	40		รหัสไปรศณีย์ในที่อยู่ใบเสร็จ	T5K2N1
Total	Float			ราคาทั้งหมด	100.00
Date_time	Datetime			เวลาในใบเสร็จ	00:00:00
Date_time_Month	Int			เดือนที่ออกใบเสร็จ	1,2,3,4
Date_time_Year	Int			ปีที่ออกใบเสร็จ	1,2,3,4
time_stamp	datetime			เวลาที่ข้อมูลเข้า	22.30 01/01/2023

3.5.4 Table : STG_INVOICES_ITEM

Description : ตารางจัดเก็บข้อมูลรายการสั่งซื้อ

ชื่อฟิลด์	ชนิดข้อมูล	ความยาว	คีย์	รายละเอียด	ตัวอย่างข้อมูล
Invt_Id	Int		PK	รหัสรายการสั่งซื้อ	1,2,3,4
Int_Id	Int		FK	รหัสใบเสร็จ	1,2,3,4
Tr_Id	Int		FK	รหัสเพลง	1,2,3,4
UnitPrice	Float			ราคา	100.00
Quantity	Int			จำนวน	1,2,3,4
time_stamp	datetime			เวลาที่ข้อมูลเข้า	22.30 01/01/2023

3.5.5 Table : STG_TRACKS

Description : ตารางจัดเก็บข้อมูลเกี่ยวกับเพลง

ชื่อฟิลด์	ชนิดข้อมูล	ความยาว	คีย์	รายละเอียด	ตัวอย่างข้อมูล
Tr_id	Int		PK	รหัสเพลง	1,2,3,4
Tr_Name	Nvarchar	200		ชื่อเพลง	Black in back
Al_id	Int		FK	รหัสอัลบัม	1,2,3,4
Med_Id	Int		FK	รหัสของชนิดไฟล์	1,2,3,4

Ge_Id	Int		FK	รหัสประเภทเพลง	1,2,3,4
Composser	Nvarchar	220		ผู้แต่ง	Augus Young
Milliseconds	Int			ความยาวของเพลง	1,2,3,4
Bytes	Int			ขนาดของเพลง	1,2,3,4
UnitPrice	Float			ราคา	100.00
time_stamp	datetime			เวลาที่ข้อมูลเข้า	22.30 01/01/2023

3.5.6 Table : STG_ALBUMS

Description : ตารางเก็บข้อมูลเกี่ยวกับอัลบ้ม

ชื่อฟิลด์	ชนิดข้อมูล	ความยาว	คีย์	รายละเอียด	ตัวอย่างข้อมูล
Al_Id	Int		PK	รหัสอัลบัม	1,2,3,4
Al_Ti	Nvarchar	160		ชื่ออัลบ้ม	Black
Ar_Id	Int		FK	รหัสศิลปิน	1,2,3,4
time_stamp	datetime			เวลาที่ข้อมูลเข้า	22.30 01/01/2023

3.5.7 Table : STG_ARTISTS

Description : ตารางเก็บข้อมูลเกี่ยวกับศิลปิน

ชื่อฟิลด์	ชนิดข้อมูล	ความยาว	คีย์	รายละเอียด	ตัวอย่างข้อมูล
Ar_Id	Int		PK	รหัสศิลปิน	1,2,3,4
Ar_Name	Nvarchar	200		ชื่อศิลปิน	AC/DC
time_stamp	datetime			เวลาที่ข้อมูลเข้า	22.30 01/01/2023

3.5.8 Table : STG_MEDIA_TYPES

Description : ตารางจัดเก็บเกี่ยวกับชนิดของไฟล์เพลง

ชื่อฟิลด์	ชนิดข้อมูล	ความยาว	คีย์	รายละเอียด	ตัวอย่างข้อมูล
Med_Id	Int		PK	รหัสของชนิดไฟล์	1,2,3,4
Me_Name	Nvarchar	120		ชนิดไฟล์	AAC audio file
time_stamp	datetime			เวลาที่ข้อมูลเข้า	22.30 01/01/2023

3.5.9 Table : STG_GENRES

Description : ตารางจัดเก็บข้อมูลเกี่ยวกับประเภทของเพลง

ชื่อฟิลด์	ชนิดข้อมูล	ความยาว	คีย์	รายละเอียด	ตัวอย่างข้อมูล
Ge_Id	Int		PK	รหัสของประเภทของเพลง	1,2,3,4
De_Name	Nvarchar	120		ประเภทของไฟล์	Metal
time_stamp	datetime			เวลาที่ข้อมูลเข้า	22.30 01/01/2023

3.5.10 Table : STG_PLAYLIST

Description : ตารางจัดเก็บข้อมูลเกี่ยวกับเพลลิสต์

ชื่อฟิลด์	ชนิดข้อมูล	ความยาว	คีย์	รายละเอียด	ตัวอย่างข้อมูล
Pl_Id	Int		PK	รหัสของเพลลิสต์	1,2,3,4
Pl_Name	Nvarchar	120		ชื่อเพลลิสต์	EiEi
time_stamp	datetime			เวลาที่ข้อมูลเข้า	22.30 01/01/2023

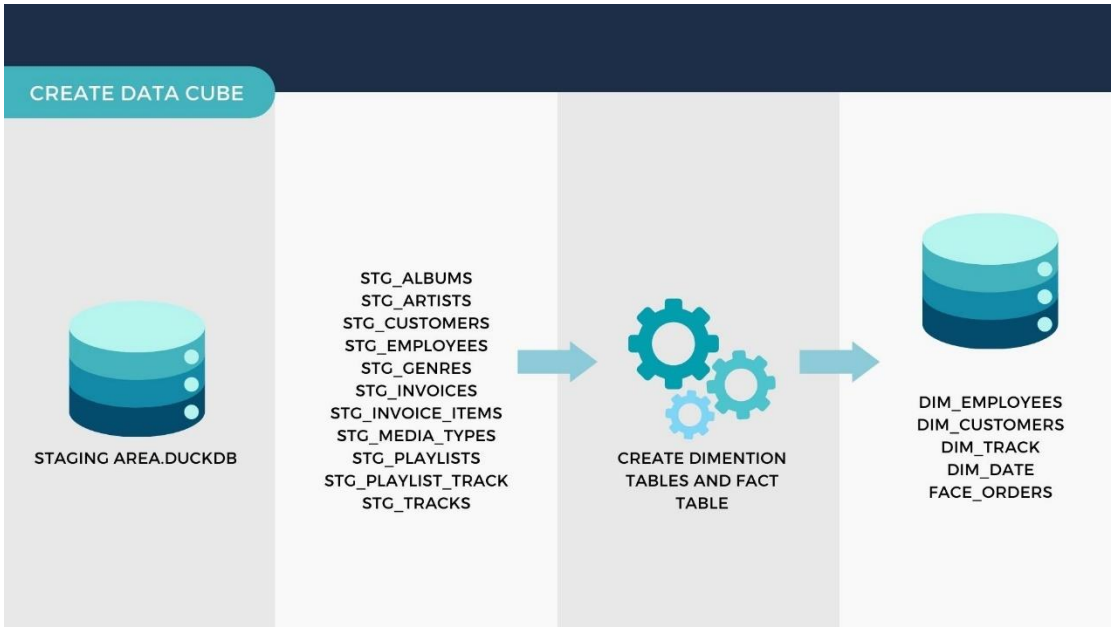
3.5.11 Table : STG_PLAYLISTS_TRACK

Description : ตารางจัดเก็บข้อมูลเกี่ยวกับเพลงที่อยู่ในเพลลิสต์

ชื่อฟิลด์	ชนิดข้อมูล	ความยาว	คีย์	รายละเอียด	ตัวอย่างข้อมูล

Pl_Id	Int		PK, FK	รหัสของเพลลิสต์	1,2,3,4
Tr_id	Int		FK	รหัสเพลง	1,2,3,4
time_stamp	Date time			เวลาที่ข้อมูลเข้า	22.30 01/01/2023

3.6 Create Data Cube



ภาพที่ 3.3 การสร้าง Data Cube

3.6.1 นำเข้า Library ที่จำเป็น

```
import polars as pl
import duckdb as dd
from datetime import datetime
import dlt
import pandas as pd
import sqlite3
```

3.6.2 เชื่อมต่อกับฐานข้อมูล Staging area database

```
# สร้างการเชื่อมต่อกับฐานข้อมูล DuckDB
db_path_op = 'operational.duckdb'
con_op = dd.connect(db_path_op)
```

3.6.3 Extraction data and close connection

```
# ดึงข้อมูลจากตารางแต่ละตาราง
Al_f = pl.read_database(query="SELECT * FROM project_stg.stg_albums",
connection=con_op)
Ar_f = pl.read_database(query="SELECT * FROM project_stg.stg_artists",
connection=con_op)
Cus_f = pl.read_database(query="SELECT * FROM project_stg.stg_customers",
connection=con_op)
Em_f = pl.read_database(query="SELECT * FROM project_stg.stg_employee",
connection=con_op)
Ge_f = pl.read_database(query="SELECT * FROM project_stg.stg_genres",
connection=con_op)
Inv_f = pl.read_database(query="SELECT * FROM project_stg.stg_invoices",
connection=con_op)
Invt_f = pl.read_database(query="SELECT * FROM project_stg.stg_invoices_items",
connection=con_op)
Med_f = pl.read_database(query="SELECT * FROM project_stg.stg_media_Types",
connection=con_op)
Pl_f = pl.read_database(query="SELECT * FROM project_stg.stg_playlist",
connection=con_op)
PlT_f = pl.read_database(query="SELECT * FROM project_stg.stg_playlist_Tracks",
connection=con_op)
Tr_f = pl.read_database(query="SELECT * FROM project_stg.stg_tracks",
connection=con_op)

# ปิดการเชื่อมต่อกับฐานข้อมูล DuckDB
con_op.close()
```

3.6.4 Create Function

```
#create Function
def rename_col(df, colname_dict):
        for old_name, new_name in colname_dict.items():
            df = df.rename({old_name: new_name})
        return df

def add_timestamp(df, colname):
        current_timestamp = datetime.now()
        df_n = df.with_columns(pl.lit(current_timestamp).alias(colname))
        return df_n

def unique(df, colname):
        return df.unique(subset=[colname])

def sort(df, colname):
        return df.sort(by=colname)

def rename_col(df, colname_dict):
        for old_name, new_name in colname_dict.items():
            df = df.rename({old_name: new_name})
        return df

def exclude(df, colname):
        return df.select(pl.col('*').exclude(colname))

def convert_str_to_datetime(df, column_name,
datetime_format='%Y-%m-%d %H:%M:%S'):
    return df.with_columns(
        pl.col(column_name).str.strptime(pl.Datetime,
format=datetime_format).alias('Date_time'))

def group_by_and_sum(df: pl.DataFrame, groupby_col: str, sum_col: str) ->
pl.DataFrame:
```

```
    return
df.group_by(groupby_col).agg(pl.col(sum_col).sum().alias(f"{sum_col}_sum"))

def group_by_and_count(df: pl.DataFrame, groupby_col: str) -> pl.DataFrame:
    return df.group_by(groupby_col).agg(pl.count().alias('count'))

def split_month_year(df, datetime_col):
    # Ensure the datetime_col is in datetime format
    df = df.with_columns(
        pl.col(datetime_col).cast(pl.Datetime).alias(datetime_col))

    # Add new columns for month and year
    return df.with_columns([
        pl.col(datetime_col).dt.month().alias(f"{datetime_col}_Month"),
        pl.col(datetime_col).dt.year().alias(f"{datetime_col}_Year")])
```

3.6.5 Transform Data and Create Data Cube

```
# สร้าง dim_tracks
Tr_select = Tr_f.select(['tr_id', 'tr_name', 'al_id', 'med_id', 'ge_id', 'composer',
'milliseconds', 'bytes', 'unit_price'])
Al_select = Al_f.select(['al_id', 'al_ti', 'ar_id'])
Med_select = Med_f.select(['med_id', 'me_name'])
Ge_select = Ge_f.select(['ge_id', 'ge_name'])
Ar_select = Ar_f.select(['ar_id', 'ar_name'])
Pl_select = Pl_f.select(['pl_id', 'pl_name'])
PlT_select = PlT_f.select(['pl_id', 'tr_id'])
Plj = Pl_select.join(PlT_select, on='pl_id', how='left')

dim_tracks = (
      Tr_select.join(Al_select, on='al_id', how='left')
      .join(Med_select, on='med_id', how='left')
      .join(Ge_select, on='ge_id', how='left')
      .join(Ar_select, on='ar_id', how='left')
      .join(Plj, on='tr_id', how='left')
      .pipe(add_timestamp,'Time_Stamp'))
```

```
print(dim_tracks)
dim_tracks.columns

# ตรวจสอบว่า address ใน Invoice และ Customer เหมือนกันหรือไม่
Inv_f_select = Inv_f.select(['cus_id', 'b_ci'])
Cus_f_select = Cus_f.select(['cus_id', 'cus_city'])
ch = Inv_f_select.join(Cus_f_select, on='cus_id', how='right')
print(ch)

# สร้าง dim_date จากข้อมูล Invoice
Inv_select = Inv_f.select(['inv_id', 'inv_d', 'date_time', 'date_time_month',
'date_time_year'])
date_inv_id = Inv_f.select(['inv_id'])

#transfrom มีการ Add timestamp
dim_date = (
    Inv_select
    .join(date_inv_id, on='inv_id', how='inner')
    .select(['inv_id', 'date_time', 'date_time_month', 'date_time_year'])
    .with_columns([
        pl.when(pl.col('date_time_month').is_between(1, 3)).then(1)
        .when(pl.col('date_time_month').is_between(4, 6)).then(2)
        .when(pl.col('date_time_month').is_between(7, 9)).then(3)
        .otherwise(4).alias('quarter')  # เพิ่มคอลัมน์ไตรมาสตามเงื่อนไข
    ])
    .pipe(add_timestamp,'Time_Stamp')
)
print(dim_date)

# สร้าง dim_customers
dim_customers = Cus_f
print(dim_customers.columns)
```

```
# สร้าง dim_employees
dim_employees = (
    Em_f
    .join(Cus_f.select(['cus_id', 'em_id']), on='em_id', how='left')  # เชื่อมกับ cus_id โดย
ใช้ em_id
)
dim_employees = dim_employees.with_columns(
    pl.when(pl.col('cus_id').is_null()).then(0).otherwise(pl.col('cus_id')).alias('cus_id')
)
dim_employees

#Fact
Inv_f.columns
Invt_f
Tr_f.columns
inv_se=Inv_f.select(['inv_id', 'cus_id', 'total'])
invt_se=Invt_f.select(['invt_id', 'inv_id', 'tr_id', 'unit_price', 'quantity'])
tr_se=Tr_f.select(['tr_id'])
date_sel=dim_date.select(['inv_id'])
cus=dim_customers.select(['cus_id'])
em_se=dim_employees.select(['cus_id','em_id'])

fact_ordaers=(inv_se.join(invt_se,on='inv_id',how='left')\
              .join(tr_se,on='tr_id',how='left')\
              .join(cus,on='cus_id',how='left')\
              .join(date_sel,on='inv_id',how='left')\
              .join(em_se,on='cus_id',how='left')
              .pipe(add_timestamp,'Time_Stamp')
)
fact_ordaers

dt=dim_tracks.to_pandas()
dc=dim_customers.to_pandas()
ddt=dim_date.to_pandas()
dem=dim_employees.to_pandas()
ftd=fact_ordaers.to_pandas()
```

3.6.6 Load data

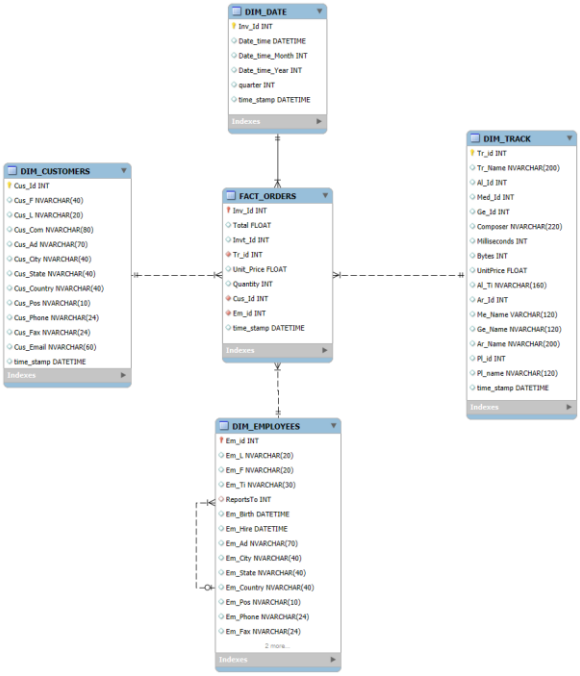
```
pipeline1=dlt.pipeline(
    pipeline_name="Dim_Fact",destination='duckdb',
    dataset_name="dimention_fact"
)

pipeline1.run(dc,table_name="dim_customers",write_disposition='append')
pipeline1.run(dem,table_name="dim_employees",write_disposition='append')
pipeline1.run(dt,table_name="dim_tracks",write_disposition='append')
pipeline1.run(ddt,table_name="dim_date",write_disposition='append')
pipeline1.run(ftd,table_name="fact_orders",write_disposition='append')
```

3.6.7 รวมตารางเป็นตาราง Data Cube ตารางเดียวเพื่อให้ง่ายต่อการดึงข้อมูลไปทำ Dashboard

```
# สร้างการเชื่อมต่อกับฐานข้อมูล DuckDB
db_path_op = 'dim_fact.duckdb'
conn1 = dd.connect(db_path_op)

date_dim = pl.read_database(query="SELECT * FROM dimention_fact.dim_date",
connection=conn1)
date_dim
em_dim = pl.read_database(query="SELECT * FROM
dimention_fact.dim_employees", connection=conn1)
em_dim
cus_dim = pl.read_database(query="SELECT * FROM
dimention_fact.dim_customers", connection=conn1)
cus_dim
tr_dim= pl.read_database(query="SELECT * FROM dimention_fact.dim_tracks",
connection=conn1)
tr_dim
fact_orders= pl.read_database(query="SELECT * FROM dimention_fact.fact_orders",
connection=conn1)
fact_orders
dd.close()
```

```
print(date_dim.columns)
print(em_dim.columns)
print(cus_dim.columns)
print(tr_dim.columns)
print(fact_orders.columns)

# เปลี่ยนชื่อ time_stamp ในแต่ละตารางก่อนทำการรวม ยกเว้นตารางหนึ่งเพื่อหลีกเลี่ยงการซ้ำกัน
tr_dim1 = tr_dim.rename({"time_stamp": "time_stamp_tr"})
cus_dim2 = cus_dim.rename({"time_stamp": "time_stamp_cus"})
em_dim3= em_dim.rename({"time_stamp": "time_stamp_em"})
date_dim4 = date_dim.rename({"time_stamp": "time_stamp_date"})

# ทำการ join ตารางอีกครั้ง
datacube = fact_orders.join(tr_dim1, on='tr_id', how='inner')\
    .join(cus_dim2, on='cus_id', how='inner')\
    .join(em_dim3, on='em_id', how='inner')\
    .join(date_dim4, on='inv_id', how='inner')
datacube

dc=datacube.to_pandas()

pipeline1.run(dc,table_name="datacube",write_disposition='append')
```

3.7 Entity-Relationship (ER) Diagram of Data cube



ภาพที่ 3.4 ER-Diagram of Data cube

3.8 Data Dictionary Entity-Relationship (ER) Diagram of Data cube

3.8.1 Table : DIM_EMPLOYEES

Description : ตารางจัดเก็บข้อมูลเกี่ยวกับมิติด้านพนักงาน

ชื่อฟิลด์	ชนิดข้อมูล	ความยาว	คีย์	รายละเอียด	ตัวอย่างข้อมูล
Em_id	Int		PK	รหัสพนักงาน	1,2,3,4
Em_L	Nvarchar	20		นามสกุลพนักงาน	Adams
Em_F	Nvarchar	20		ชื่อจริงพนักงาน	Andrew
Em_Ti	Nvarchar	30		ชื่อตำแหน่งพนักงาน	General Manager

ReportsTo	Int		FK	รหัสหัวหน้าของพนักงาน	1,2,3,4
Em_Birth	Datetime			วันเกิดพนักงาน	1962-02-18 00:00:00
Em_Hire	Datetime			วันที่จ้างพนักงาน	2002-08-14 00:00:00
Em_Ad	Nvarchar	70		ที่อยู่ของพนักงาน	11120 Jasper Ave NW
Em_City	Nvarchar	40		เมืองของพนักงาน	Edmonton
Em_State	Nvarchar	40		รัฐของพนักงาน	AB
Em_Country	Nvarchar	40		ประเทศของพนักงาน	Canada
Em_Pos	Nvarchar	10		รหัสไปรษณีย์ของพนักงาน	T5K2N1
Em_Phone	Nvarchar	24		เบอร์โทรศัพท์ของพนักงาน	+1(780)428-9482
Em_Fax	Nvarchar	24		แฟกซ์ของพนักงาน	+1(780)428-3457
Em_Email	Nvarchar	60		อีเมลของพนักงาน	andrew@chinookcorp.com
time_stamp	datetime			เวลาที่ข้อมูลเข้า	22.30 01/01/2023

3.8.2 Table : DIM_CUSTOMERS

Description : ตารางจัดเก็บข้อมูลเกี่ยวกับมิติในด้านลูกค้า

ชื่อฟิลด์	ชนิดข้อมูล	ความยาว	คีย์	รายละเอียด	ตัวอย่างข้อมูล
Cus_id	Int		PK	รหัสลูกค้า	1,2,3,4
Cus_F	Nvarchar	40		ชื่อจริงของลูกค้า	Adams
Cus_L	Nvarchar	20		นามสกุลของลูกค้า	Andrew
Cus_Com	Nvarchar	80		บริษัทของลูกค้า	JetBrains s.r.o.
Cus_Ad	Nvarchar	70		ที่อยู่ของลูกค้า	Theodor-Heuss-Straße 34
Cus_City	Nvarchar	40		เมืองของลูกค้า	Edmonton
Cus_State	Nvarchar	40		รัฐของลูกค้า	AB
Cus_Country	Nvarchar	40		ประเทศของลูกค้า	Canada
Cus_Pos	Nvarchar	10		รหัสไปรษณีย์ของลูกค้า	T5K2N1
Cus_Phone	Nvarchar	24		เบอร์โทรศัพท์ของลูกค้า	+1(780)428-9482
Cus_Fax	Nvarchar	24		แฟกซ์ของลูกค้า	+1(780)428-3457
Cus_Email	Nvarchar	60		อีกเมลของลูกค้า	luisg@embraer.com.br

Em_id	Int		FK	รหัสของพนักงานที่ทำการดูแล	1,2,3,4
time_stamp	datetime			เวลาที่ข้อมูลเข้า	22.30 01/01/2023

3.8.3 Table : TRACKS

Description : ตารางจัดเก็บข้อมูลเกี่ยวกับเพลง

ชื่อฟิลด์	ชนิดข้อมูล	ความยาว	คีย์	รายละเอียด	ตัวอย่างข้อมูล
Tr_id	Int		PK	รหัสเพลง	1,2,3,4
Tr_Name	Nvarchar	200		ชื่อเพลง	Black in back
Al_id	Int		FK	รหัสอัลบัม	1,2,3,4
Med_Id	Int		FK	รหัสของชนิดไฟล์	1,2,3,4
Ge_Id	Int		FK	รหัสประเภทเพลง	1,2,3,4
Composser	Nvarchar	220		ผู้แต่ง	Augus Young
Milliseconds	Int			ความยาวของเพลง	1,2,3,4
Bytes	Int			ขนาดของเพลง	1,2,3,4
UnitPrice	Float			ราคา	100.00
Al_Ti	Nvarchar	160		ชื่ออัลบัม	EiEi

Ar_Id	Int			รหัสศิลปิน	1,2,3,4
Me_Name	Nvarchar	120		ชนิดไฟล์	AAC
Ge_Name	Nvarchar	120		ชื่อประเภท	Movies
Ar_Name	Nvarchar	200		ชื่อศิลปิน	AC/DC
Pl_id	Int			รหัสเพลลิสต์	1,2,3,4
Pl_name	Nvarchar	120		ชื่อเพลลิสต์	EiEi
time_stamp	datetime			เวลาที่ข้อมูลเข้า	22.30 01/01/2023

3.8.4 Table : DIM_DATE

Description : ตารางจัดเก็บข้อมูลเกี่ยวกับมิติด้านเวลา

ชื่อฟิลด์	ชนิดข้อมูล	ความยาว	คีย์	รายละเอียด	ตัวอย่างข้อมูล
Inv_Id	Int		PK	รหัสใบเสร็จ	1,2,3,4
Date_time	Datetime			วันเวลา	2024-08-11 00:00:00
Date_time_Month	Int			เดือน	8

Date_time_ Year	Int			ปี	2024
quarter	Int			ไตรมาสต์	1,2,3,4
time_stamp	datetime			เวลาที่ข้อมูลเข้า	22.30 01/01/2023

3.8.5 Table : FACT_ORDERS

Description : ตารางจัดเก็บข้อมูลเกี่ยวกับรายการที่ใช้คำนวณ

ชื่อฟิลด์	ชนิดข้อมูล	ความยาว	คีย์	รายละเอียด	ตัวอย่างข้อมูล
Inv_Id	Int		PK	รหัสใบเสร็จ	1,2,3,4
Total	Float			ราคาทั้งหมด	100.00
Invt_Id	Int			รหัสรายการสั่งซื้อ	1,2,3,4
Tr_id	Int			รหัสเพลง	1,2,3,4
UnitPrice	Float			ราคา	100.00
Quantity	Int			จำนวน	1,2,3,4
Cus_Id	Int			รหัสลูกค้า	1,2,3,4
Em_Id	Int			รหัสพนักงาน	1,2,3,4
time_stamp	datetime			เวลาที่ข้อมูลเข้า	22.30 01/01/2023

3.9 การสร้าง Dashboard

3.9.1 Import Libraries และการเชื่อมต่อกับฐานข้อมูล

```
import polars as pl
import streamlit as st
import pandas as pd
import duckdb as dd
import plotly.express as px
from streamlit_echarts import st_echarts

# ดึงข้อมูลจาก database ``
conn = dd.connect(r'C:\DataWarehouse_Project\dim_fact_dashboard.duckdb')
datacube = pl.read_database(
    query= "SELECT * FROM dimention_fact.datacube",
    connection=conn )
conn.close()
```

3.9.2 การตั้งค่า Streamlit และ Sidebar สำหรับเลือกเพจ

```
st.title('Music Stores Data Cube Viewer Dashboard')


page = st.sidebar.selectbox("Select Page", ["Sales Dashboard", "Heat Map", "Employee Sales"])
```

3.9.3 หน้า "Sales Dashboard"

```


if page == "Sales Dashboard":
    # ทำ aggregation ของข้อมูลของยอดขายของแต่ละประเทศ
    country_sales = datacube.group_by(['cus_contry']).agg([
        pl.col('total').sum().alias('total_sales')
    ]).sort(['total_sales'], descending=[True]).pipe(rename_col, {'cus_contry': 'Country', 'total_sales': 'Total Sales'}).to_pandas()
    df = pd.read_csv(r'C:\DataWarehouse_Project\all.csv')
```

```
    country_sales = country_sales.merge(df[['name', 'region']], left_on='Country',
right_on='name', how='left')
    country_sales = country_sales.drop(columns=['name'])
    regions = {
        'USA': 'Americas',
        'United Kingdom': 'Europe',
        'Czech Republic': 'Europe',
        'Netherlands': 'Europe'
    }
    country_sales['region'] =
country_sales['region'].fillna(country_sales['Country'].map(regions))

    st.sidebar.header('Filter Options')
    region_filter = st.sidebar.multiselect('Select Region(s):',
options=country_sales['region'].unique(), default=country_sales['region'].unique())

    filtered_countries =
country_sales[country_sales['region'].isin(region_filter)]['Country'].unique()

    country_filter = st.sidebar.multiselect('Select Country(s):',
options=filtered_countries, default=filtered_countries)

    filtered_data = country_sales[
    (country_sales['region'].isin(region_filter)) &
    (country_sales['Country'].isin(country_filter))
    ]

    data = filtered_data.groupby(['region', 'Country']).sum().reset_index()

    sunburst_sales = px.sunburst(
    data,
    path=['region', 'Country'],
    values='Total Sales',
    title='Sales by Region, Country'
    )
# แยก 2 ฝั่งของ layout ให้เป็น 2 คอลัมน์
```

```
    col1, col2 = st.columns(2)
# แสดง sunburst chart ในคอลัมน์แรก
    with col1:
        st.subheader('Sales by Region, Country (Sunburst Chart)')
        st.plotly_chart(sunburst_sales, use_container_width=True)
# แสดง bar chart ในคอลัมน์ที่สอง
    with col2:
        st.subheader('Total Sales by Region and Country (Stacked Bar Chart)')
        stacked_bar_chart = px.bar(
            filtered_data,
            x='Country',
            y='Total Sales',
            color='region',
            title='Total Sales by Region and Country',
            labels={'Total Sales': 'Total Sales ($)', 'Country': 'Country'},
            barmode='stack'
        )
        st.plotly_chart(stacked_bar_chart, use_container_width=True)

    # เลือกเฉพาะข้อมูลที่มี region และ country ตามที่เลือก
    filtered_datacube = datacube.filter(pl.col('cus_contry').is_in(country_filter))
    # ทำ aggregation ของข้อมูลของยอดขายของแต่ละประเทศ
    yearly_country_sales = filtered_datacube.group_by(['date_time_year',
'cus_contry']).agg([
        pl.col('total').sum().alias('total_sales')
    ]).sort(['date_time_year', 'total_sales'], descending=[False,
True]).pipe(rename_col, {'date_time_year': 'Year', 'cus_contry': 'Country',
'total_sales': 'Total Sales'})

    yearly_country_sales_df = yearly_country_sales.to_pandas()

    # สร้าง line chart
    yearly_line_chart = px.line(
        yearly_country_sales_df,
        x='Year',
        y='Total Sales',
```

```
        color='Country',
        title='Yearly Sales by Country',
        labels={'Total Sales': 'Total Sales ($)', 'Year': 'Year'},
        line_group='Country'
    )

    month_sales = filtered_datacube.group_by(['date_time_month',
'cus_contry']).agg([
        pl.col('total').sum().alias('total_sales')
    ]).sort(['date_time_month', 'total_sales'],
descending=[False,True]).pipe(rename_col,
    {'date_time_month': 'Month', 'cus_contry': 'Country', 'total_sales': 'Total Sales'})

    monthly_sales_df = month_sales.to_pandas()

    #สร้าง line chart สำหรับยอดขายรายเดือน
    stacked_line_chart = px.line(
        monthly_sales_df,
        x='Month',
        y='Total Sales',
        color='Country',
        title='Monthly Sales by Country',
        labels={'Total Sales': 'Total Sales ($)', 'Month': 'Month'},
        line_group='Country'
    )

    col3, col4 = st.columns(2)
# แสดง line chart สำหรับยอดขายรายเดือนในคอลัมน์แรก
    with col3:
        st.subheader('Monthly Sales by Country (Stacked Line Chart)')
        st.plotly_chart(stacked_line_chart, use_container_width=True)
# แสดง line chart สำหรับยอดขายรายปีในคอลัมน์ที่สอง
    with col4:
        st.subheader('Yearly Sales by Country (Line Chart)')
        st.plotly_chart(yearly_line_chart, use_container_width=True)
```

3.9.4 Heat Map Page

```
elif page == "Heat Map":
    genre_sales = datacube.group_by(['date_time_year','ge_name']).agg([
        pl.col('total').sum().alias('total_sales')
    ]).sort(['date_time_year','total_sales'], descending=[False,True]).pipe(rename_col,
{'date_time_year': 'Year', 'ge_name': 'Genre', 'total_sales': 'Total Sales'})
    st.header('Heat Map Genre Sales')
    heatmap_data = []
    years = genre_sales['Year'].unique().to_list()
    genres = genre_sales['Genre'].unique().to_list()

    for year in years:
        for genre in genres:
            total_sales = genre_sales.filter((pl.col('Year') == year) & (pl.col('Genre') ==
genre))['Total Sales']
            if total_sales.is_empty():
                value = 0
            else:
                value = total_sales[0]
            heatmap_data.append([genres.index(genre), years.index(year), value])

    option_heatmap = {
        "title": {
            "text": 'Genre Sales Heatmap',
            "left": 'center'
        },
        "tooltip": {
            "position": 'top'
        },
        "grid": {
            "height": '50%',
            "top": '10%'
        },
        "xAxis": {
            "type": 'category',
            "data": genres,
```

```
        "splitArea": {
            "show": True
        },
        "axisLabel": {
            "interval": 0,
            "rotate": 75
        }
    },
    "yAxis": {
        "type": 'category',
        "data": years,
        "splitArea": {
            "show": True
        }
    },
    "visualMap": {
        "min": 0,
        "max": max([item[2] for item in heatmap_data]),
        "calculable": True,
        "orient": 'horizontal',
        "left": 'center',
        "bottom": '15%'
    },
    "series": [{
        "name": 'Total Sales',
        "type": 'heatmap',
        "data": heatmap_data,
        "label": {
            "show": False
        },
        "emphasis": {
            "itemStyle": {
                "shadowBlur": 10,
                "shadowColor": 'rgba(0, 0, 0, 0.5)'
            }
        }
```

```
        }]
    }

    st_echarts(option_heatmap, height="800px", width="100%")
```

3.9.5 Employee Sales Page

```
elif page == "Employee Sales":
    top_employee_sales = datacube.group_by(['date_time_year','em_f','em_l']).agg([
        pl.col('total').sum().alias('total_sales')
    ]).sort(['date_time_year','total_sales'], descending=[False,True]).pipe(rename_col,
{'date_time_year': 'Year', 'em_f': 'First', 'em_l': 'Last', 'total_sales': 'Total
Sales'}).to_pandas()
    st.header('Top Employee Sales')
    steve_sales = top_employee_sales[top_employee_sales['First'] == 'Steve']['Total
Sales'].to_list()
    jane_sales = top_employee_sales[top_employee_sales['First'] == 'Jane']['Total
Sales'].to_list()
    margaret_sales = top_employee_sales[top_employee_sales['First'] ==
'Margaret']['Total Sales'].to_list()

    options_mix = {
        "tooltip": {
            "trigger": "axis",
            "axisPointer": {"type": "cross", "crossStyle": {"color": "#999"}},
        },
        "toolbox": {
            "feature": {
                "dataView": {"show": True, "readOnly": False},
                "magicType": {"show": True, "type": ["line", "bar"]},
                "restore": {"show": True},
                "saveAsImage": {"show": True},
            }
        },
        "legend": {"data": ["Steve Johnson", "Jane Peacock", "Margaret Park", "Total
Sales"]},
```

```
    "xAxis": [
        {
            "type": "category",
            "data": top_employee_sales['Year'].unique().tolist(),
            "axisPointer": {"type": "shadow"},
        }
    ],
    "yAxis": [
        {
            "type": "value",
            "name": "Sales",
            "min": 0,
            "axisLabel": {"formatter": "{value} $"},
        },
    ],
    "series": [
        {
            "name": "Steve Johnson",
            "type": "bar",
            "data": steve_sales,
        },
        {
            "name": "Jane Peacock",
            "type": "bar",
            "data": jane_sales,
        },
        {
            "name": "Margaret Park",
            "type": "bar",
            "data": margaret_sales,
        },
        {
            "name": "Total Sales",
            "type": "line",
            "data": top_employee_sales.groupby('Year')['Total Sales'].sum().tolist(),
        },
```

```
    ],
}
st_echarts(options_mix, height="400px", width="100%")
```

บทที่ 4

ผลการดำเนินโครงงาน

จากการดำเนินโครงงานเกี่ยวกับการออกแบบ Multidimensional Data Model และการพัฒนา Web Application เพื่อการวิเคราะห์ข้อมูล ได้ผลการศึกษา ดังนี้

4.1 การออกแบบ Multidimensional Data Model

การออกแบบฐานข้อมูลในรูปแบบของ Star Schema ได้ช่วยเพิ่มประสิทธิภาพในการวิเคราะห์ข้อมูลเชิงลึก โดยเฉพาะในบริบทของการวิเคราะห์ข้อมูลการขายในระบบธุรกิจที่มีหลายมิติ เช่น การวิเคราะห์ข้อมูลตามช่วงเวลา ประเภทสินค้า ลูกค้า พนักงาน และแนวเพลงที่ขาย โดย Fact Table และ Dimension Tables ที่ถูกสร้างขึ้นได้ครอบคลุมทุกมิติที่เกี่ยวข้อง ซึ่งทำให้สามารถเข้าถึงข้อมูลได้อย่างมีประสิทธิภาพและง่ายต่อการใช้งานสำหรับการวิเคราะห์ข้อมูลเชิงธุรกิจในอนาคต

ข้อมูลใน Fact Table ถูกออกแบบให้สามารถเก็บข้อมูลเชิงปริมาณ เช่น ยอดขาย จำนวนสินค้า และราคาต่อหน่วย ซึ่งทำให้สามารถวิเคราะห์ข้อมูลในระดับสถิติได้อย่างละเอียด

Dimension Tables ที่ออกแบบมาครอบคลุมมิติต่าง ๆ ของธุรกิจ เช่น ลูกค้า พนักงาน เพลง และวันที่ ทำให้สามารถวิเคราะห์ข้อมูลเชิงลึกในหลายแง่มุม เช่น การขายตามช่วงเวลา ประสิทธิภาพการทำงานของพนักงาน หรือยอดขายที่มาจากลูกค้ากลุ่มต่าง ๆ

4.2 การพัฒนา Web Application

การพัฒนา Web Application สำหรับการแสดงผลข้อมูลได้ช่วยให้สามารถนำข้อมูลจากฐานข้อมูลไปใช้ในการวิเคราะห์และรายงานผลได้อย่างมีประสิทธิภาพ ตัวแอปพลิเคชันถูกออกแบบให้สามารถแสดงข้อมูลในรูปแบบของ Dashboard ซึ่งเป็นเครื่องมือสำคัญในการติดตามและวิเคราะห์ข้อมูลเชิงธุรกิจ โดย Web Application นี้มีคุณสมบัติ ดังนี้

4.2.1 แสดงผลข้อมูลในรูปแบบของกราฟและตารางที่สามารถปรับแต่งได้ตามความต้องการของผู้ใช้งาน

4.2.2 มีระบบการค้นหาและกรองข้อมูลที่ช่วยให้ผู้ใช้สามารถเข้าถึงข้อมูลเฉพาะเจาะจงตามช่วงเวลาหรือกลุ่มลูกค้าได้อย่างง่ายดาย

4.2.3 เชื่อมต่อกับฐานข้อมูล Multidimensional Data Model โดยตรง ทำให้สามารถดึงข้อมูลออกมาแสดงผลได้แบบ Real-time

จากการดำเนินโครงงาน สามารถสรุปได้ว่า Multidimensional Data Model และ Web Application ที่ถูกพัฒนาขึ้นสามารถช่วยเพิ่มประสิทธิภาพในการวิเคราะห์และการจัดการข้อมูลเชิงธุรกิจได้อย่างมาก และเป็นเครื่องมือที่มีความสำคัญสำหรับการตัดสินใจเชิงกลยุทธ์ในอนาคต

บทที่ 5

สรุปผลการดำเนินโครงงาน ปัญหา และข้อเสนอแนะ

5.1 ผลการดำเนินโครงงาน

จากการดำเนินโครงงานนี้ เราได้ทำการวิเคราะห์ข้อมูลยอดขายเพลงโดยใช้ฐานข้อมูล Chinook ผ่านการสร้างมุมมองธุรกิจที่หลากหลาย เพื่อทำความเข้าใจภาพรวมของตลาดเพลงในแต่ละแง่มุม นอกจากนี้ยังได้พัฒนาโมเดลข้อมูลแบบ Multidimensional และนำเสนอข้อมูลผ่านแดชบอร์ด เพื่อให้เห็นภาพรวมและรายละเอียดในเชิงลึกของข้อมูลได้อย่างชัดเจน ผลลัพธ์จากโครงงานนี้สามารถสรุปได้ดังนี้

5.1.1 ได้มุมมองทางธุรกิจที่ครอบคลุมทั้งในด้านยอดขายของเพลง ยอดขายในแต่ละภูมิภาค ผลงานของพนักงาน และความสำเร็จของศิลปิน

5.1.2 สร้างโมเดลข้อมูลแบบ Multidimensional ที่สามารถรองรับการวิเคราะห์ข้อมูลในเชิงลึก โดยเฉพาะการวิเคราะห์ยอดขายตามมิติธุรกิจที่สนใจ เช่น เวลา ศิลปิน ประเภทเพลง และพนักงาน

แดชบอร์ดที่พัฒนาขึ้นสามารถแสดงข้อมูลได้ในรูปแบบที่เข้าใจง่าย มีความยืดหยุ่นในการปรับปรุงเพื่อใช้ในงานวิเคราะห์ต่อไป

5.2 ปัญหา

ในการดำเนินโครงงาน พบปัญหาหลายประการที่ต้องแก้ไขและปรับปรุงในกระบวนการพัฒนา ตั้งแต่การกำหนดมุมมองธุรกิจ การสร้างโมเดลข้อมูล จนถึงการสร้างแดชบอร์ดเพื่อนำเสนอข้อมูล

5.2.1 ปัญหาที่พบในการกำหนดมุมมองธุรกิจ

5.2.1.1 การเลือกมุมมองที่มีความสำคัญต่อธุรกิจ

การตัดสินใจเลือกมุมมองธุรกิจบางครั้งพบว่าไม่สามารถเจาะลึกข้อมูลที่ต้องการได้อย่างเต็มที่ เนื่องจากการเลือกมุมมองในบางครั้งอาจไม่สอดคล้องกับลักษณะข้อมูลที่มีอยู่ในฐานข้อมูล Chinook ซึ่งทำให้ต้องปรับแก้มุมมองเพื่อให้เข้ากับข้อมูลจริงที่ใช้ในการวิเคราะห์

5.2.1.2 ข้อจำกัดของฐานข้อมูล: ฐานข้อมูล Chinook มีขอบเขตข้อมูลที่จำกัดอยู่ในด้านศิลปิน เพลง และการขายดิจิทัล ซึ่งทำให้ไม่สามารถเพิ่มมุมมองที่เกี่ยวกับการขายในตลาดอื่น ๆ ได้ เช่น ตลาดเพลงสตรีมมิ่งหรือการจัดจำหน่ายแบบฟิสิคัล (physical distribution)

5.2.2 ปัญหาที่พบในการสร้าง Mulitdimensional data model

5.2.2.1 ความซับซ้อนในการเชื่อมโยงข้อมูลหลายมิติ

การออกแบบโมเดลข้อมูลแบบหลายมิติ (Multidimensional Data Model) ต้องการการทำความเข้าใจความสัมพันธ์ระหว่างตารางต่าง ๆ อย่างละเอียด ซึ่งในบางครั้งพบปัญหาในการเชื่อมโยงข้อมูลที่มีโครงสร้างซับซ้อน โดยเฉพาะในส่วนที่ต้องรวมข้อมูลหลายมิติเพื่อใช้ในการวิเคราะห์ข้อมูลเชิงลึก

5.2.2.2 การจัดการข้อมูลขนาดใหญ่

เมื่อข้อมูลมีขนาดใหญ่ขึ้น การสร้างโมเดลแบบ Multidimensional ที่มีประสิทธิภาพอาจทำได้ยาก เนื่องจากข้อจำกัดทางเทคนิคในการจัดการข้อมูลขนาดใหญ่และการเพิ่มประสิทธิภาพการประมวลผล

5.2.2.3 ข้อมูลเดียวกันมีตัวแปรต่างกัน

ในการสร้าง Multidimensional Data Model ในครั้งนี้ มีปัญหาในเรื่องของการเตรียมข้อมูล อยู่หลายประเด็น ข้อมูลที่เก็บค่าเดียวกันเมื่ออยู่คนละตาราง มีการตั้งชื่อที่ต่างกัน และไม่มีการอธิบายข้อมูลในส่วนนี้ ทำให้ต้องมีการเรียกดูข้อมูลเพื่อนำมาตรวจสอบหาข้อสรุป และมีหลายตารางที่เก็บข้อมูลต่างกันแต่ตั้งชื่อเหมือนกัน ต้องมีการเปลี่ยนชื่อฟิลด์หลายตาราง และต้องมีการตรวจเช็คความถูกต้อง ความเรียบร้อยของข้อมูลหลายครั้ง

5.2.3 ปัญหาที่พบในการสร้างแดชบอร์ดนำเสนอข้อมูลของฐานข้อมูล Chinook

5.2.3.1 ความท้าทายในการออกแบบ UI/UX

การออกแบบแดชบอร์ดที่สามารถแสดงข้อมูลได้อย่างชัดเจนและเข้าใจง่ายถือเป็นความท้าทาย โดยเฉพาะการจัดเรียงข้อมูลและการใช้กราฟิกที่สื่อความหมายได้ตรงประเด็นและตอบโจทย์ผู้ใช้งาน

5.2.3.2 ความยืดหยุ่นในการปรับเปลี่ยนข้อมูล

แดชบอร์ดบางส่วนอาจมีข้อจำกัดในการปรับเปลี่ยนมุมมองหรือกรองข้อมูลในแบบเรียลไทม์ เนื่องจากโครงสร้างข้อมูลที่ถูกกำหนดไว้ล่วงหน้า ทำให้บางครั้งผู้ใช้ไม่สามารถเปลี่ยนแปลงหรือเจาะลึกข้อมูลในบางด้านได้ตามต้องการ

5.3 ข้อเสนอแนะ

จากการดำเนินโครงงานนี้ มีข้อเสนอแนะสำหรับการปรับปรุงและพัฒนาต่อไปในอนาคต ดังนี้

5.3.1 การปรับปรุงมุมมองธุรกิจ

ควรมีการศึกษาความต้องการของผู้ใช้งานและตลาดอย่างต่อเนื่อง เพื่อให้สามารถเลือกมุมมองธุรกิจที่สอดคล้องกับความต้องการและข้อมูลจริงได้ดียิ่งขึ้น รวมถึงการพิจารณาการใช้แหล่งข้อมูลที่ครอบคลุมกว่าฐานข้อมูล Chinook เพื่อเพิ่มโอกาสในการวิเคราะห์ข้อมูลที่กว้างขวางและลึกซึ้งมากยิ่งขึ้น เช่น การเพิ่มข้อมูลจากแหล่งอื่น ๆ เกี่ยวกับการสตรีมเพลง การจำหน่ายแบบฟิสิคัล หรือข้อมูลจากแพลตฟอร์มดิจิทัลต่าง ๆ

5.3.2 การออกแบบโมเดลข้อมูลที่ยืดหยุ่น

ควรปรับปรุงโครงสร้างโมเดล Multidimensional Data Model ให้มีความยืดหยุ่นมากขึ้นในการเชื่อมโยงข้อมูลและรองรับข้อมูลขนาดใหญ่ การใช้เทคนิคการเพิ่มประสิทธิภาพการประมวลผลและการเลือกเครื่องมือที่เหมาะสมจะช่วยลดความซับซ้อนในการจัดการข้อมูลขนาดใหญ่ อีกทั้งควรมีการทดสอบประสิทธิภาพการเชื่อมโยงข้อมูลเพื่อให้มั่นใจว่าโมเดลสามารถใช้งานได้อย่างรวดเร็วและมีประสิทธิภาพสูง

5.3.3 การปรับปรุงแดชบอร์ดเพื่อการใช้งานจริง

ในการออกแบบแดชบอร์ด ควรมีการวิจัยด้าน UI/UX เพื่อให้แดชบอร์ดตอบโจทย์การใช้งานจริงและสื่อสารข้อมูลได้อย่างมีประสิทธิภาพ การเพิ่มความยืดหยุ่นให้กับผู้ใช้งานในการปรับมุมมองและกรองข้อมูลได้เองในแบบเรียลไทม์จะช่วยให้แดชบอร์ดเป็นเครื่องมือที่ใช้งานได้จริงและมีประโยชน์สูงสุดในการตัดสินใจธุรกิจ

5.3.4 การพัฒนาและปรับปรุงเทคโนโลยีที่ใช้

ควรพิจารณาการใช้เทคโนโลยีใหม่ ๆ ที่มีประสิทธิภาพสูงในการจัดการและวิเคราะห์ข้อมูล เช่น การใช้ระบบคลังข้อมูล (Data Warehouse) ที่รองรับการวิเคราะห์ข้อมูลในระดับใหญ่ และการใช้เครื่องมือ BI (Business Intelligence) ที่มีความยืดหยุ่นในการวิเคราะห์และนำเสนอข้อมูล

5.3.5 การพัฒนาเพิ่มเติมจากผลการวิเคราะห์

ควรใช้ผลลัพธ์ที่ได้จากการวิเคราะห์ข้อมูลในแดชบอร์ดมาปรับใช้ในเชิงกลยุทธ์ เพื่อเพิ่มประสิทธิภาพทางธุรกิจ เช่น การปรับเปลี่ยนกลยุทธ์การขาย การปรับปรุงผลิตภัณฑ์เพลง หรือการพัฒนาทีมพนักงานตามผลการวิเคราะห์ที่ได้จากข้อมูลยอดขาย

เอกสารอ้างอิง

1. IBM. (n.d.). **What is ETL?.**สืบค้นเมื่อ 10 สิงหาคม 2567, จาก https://www.ibm.com /topics/etl
2. Kimball, R., & Ross, M. (2013). **The data warehouse toolkit: The definitive guide to dimensional modeling (ฉบับที่ 3).** John Wiley & Sons, Inc.
3. Kodjin. (n.d.). **ETL process in the healthcare industry.** สืบค้นเมื่อ 10 สิงหาคม 2567, จาก https://kodjin.com/blog/etl-process-in-the-healthcare-industry/
4. NT Cloud Solutions. (n.d.). **Data warehouse คืออะไร?.** สืบค้นเมื่อ 15 สิงหาคม 2567, จาก https://ntcloudsolutions.ntplc.co.th/knowledge/data-warehouse/
5. Rainardi, V. (2008). **Building a data warehouse: With examples in SQL Server. Apress.** https://doi.org/10.1007/978-1-4302-0527-2
6. SCB Tech X. (ม.ป.ป.). **Data warehouse คืออะไร? ไม่อยากตกเทรนด์ Data ต้องรู้เอาไว้!.** สืบค้นเมื่อ 15 สิงหาคม 2567, จาก https://scbtechx.io/th/blogs/what-is-a-data-warehouse/
7. SQLite Tutorial. (n.d.). **SQLite sample database.** สืบค้นเมื่อ 30 กรกฎาคม 2024, จาก https://www.sqlitetutorial.net/sqlite-sample-database/
8. Wikipedia contributors. (n.d.). **Data warehouse.** สืบค้นเมื่อ 16 สิงหาคม 2024, จาก https://en.wikipedia.org/wiki/Data_warehouse
9. Wikipedia contributors. (n.d.). **Staging area.** สืบค้นเมื่อ 16 สิงหาคม 2024, จาก https://en.wikipedia.org/wiki/Staging_area