ระบบฐานข้อมูลสำหรับการซื้อขายสินค้าในร้านขายยา

นางสาวเมตตา เทียนชนะไชยา

การศึกษาโครงการเฉพาะเรื่องนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตร
ปริญญาวิทยาศาสตรมหาบัณฑิต สาขาวิชาเทคโนโลยีสารสนเทศ
คณะเทคโนโลยีสารสนเทศ
มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าธนบุรี
พ.ศ. 2556

ระบบฐานข้อมูลสำหรับการซื้อขายสินค้าในร้านขายยา

นางสาวเมตตา เทียนชนะไชยา วศ.บ. (อิเล็กทรอนิกส์และโทรคมนาคม)

การศึกษาโครงการเฉพาะเรื่องนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตร
ปริญญาวิทยาศาสตรมหาบัณฑิต สาขาวิชาเทคโนโลยีสารสนเทศ
คณะเทคโนโลยีสารสนเทศ
มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าธนบุรี
พ.ศ. 2556

คณะกรรมการการศึกษาโครงการเฉพาะเรื่อง

..... ประธานกรรมการการศึกษาโครงการเฉพาะเรื่อง
(ดร.สุรีย์ ฟูนิลกุล)

..... กรรมการและอาจารย์ที่ปรึกษาร่วม
(ผศ. สุเมธ อังคะศิริกุล)

..... กรรมการ
(รศ. ดร.วิเชียร ชุติมาสกุล)

หัวข้อการศึกษาโครงการเฉพาะเรื่อง	ระบบฐานข้อมูลสำหรับการซื้อขายสินค้าในร้านขายยา
หน่วยกิต	3
ผู้เขียน	นางสาวเมตตา เทียนชนะไชยา
อาจารย์ที่ปรึกษา	ดร.สุรีย์ ฟูนิลกุล
	ผศ. สุเมธ อังคะศิริกุล
หลักสูตร	วิทยาศาสตรมหาบัณฑิต
สาขาวิชา	เทคโนโลยีสารสนเทศ
คณะ	เทคโนโลยีสารสนเทศ
พ.ศ.	2556

บทคัดย่อ

การศึกษาโครงการเฉพาะเรื่องนี้ได้ศึกษาการพัฒนาระบบฐานข้อมูลเพื่อใช้ในร้านขายยา มีวัตถุประสงค์เพื่อศึกษาการออกแบบ การสร้าง และการจัดการข้อมูลในฐานข้อมูล จากนั้นนำความรู้ที่ได้ศึกษาและออกแบบมาทำการสร้างฐานข้อมูลจริงโดยใช้ซอฟต์แวร์ Oracle Database 11g เพื่อใช้ในการจัดเก็บข้อมูลในร้านขายยา อีกทั้งยังสามารถเรียกดูและออกรายงานการขายสินค้าภายในร้านได้โดยใช้เครื่องมือ SQL Developer ผลการศึกษาในการพัฒนาฐานข้อมูลนั้นสามารถบันทึกข้อมูลการซื้อขายและข้อมูลสินค้าได้ สามารถเรียกดูตำแหน่งของสินค้า เรียกดูสินค้าที่หมดในคลังสินค้า เรียกดูยอดเงินรวมต่อ 1ใบเสร็จรับเงิน เรียกดูยอดเงินรวมของการขายสินค้าในแต่ละวัน เรียกดูสินค้าที่ขายในแต่ละวัน เรียกดูรายละเอียดการขายต่อ 1 ใบเสร็จรับเงิน และเรียกดูลำดับสินค้าที่ขายได้มากที่สุดไปหาน้อยสุดได้ เพื่อเพิ่มประสิทธิภาพในการจัดเก็บข้อมูลในร้านขายยาให้มีความเป็นระบบ และยังนำข้อมูลที่ได้จากการค้นหาในฐานข้อมูลร้านขายยามาช่วยสนับสนุนการตัดสินใจ ใช้ในการวิเคราะห์วางแผน และพัฒนากลยุทธ์ทางการตลาดเพื่อเพิ่มความได้เปรียบทางธุรกิจร้านขายยา

คำสำคัญ : ระบบฐานข้อมูล / ร้านขายยา / ออราเคิล 11 จี / เอสคิวแอล_เดเวลอปเปอร์

สารบัญ

สารบัญ (ต่อ)

รายการตารางประกอบ

รายการรูปประกอบ

รายการรูปประกอบ (ต่อ)

บทที่ 1 บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

ร้านขายยาเป็นการประกอบธุรกิจขายสินค้าประเภทยาและสินค้าที่เกี่ยวกับสุขอนามัยอื่น ๆ โดยมีเภสัชกรเป็นผู้ให้บริการขายสินค้าประเภทยาตามความต้องการของลูกค้า แต่เนื่องจากร้านขายยาในปัจจุบันขายสินค้าหลากหลายประเภท ทำให้ยากต่อการจัดการสินค้าภายในร้าน และยังส่งผลให้เกิดความผิดพลาดต่าง ๆ ได้ เช่น การคิดคำนวณราคาสินค้ามีความล่าช้า การหาสินค้าภายในร้านไม่พบ จำนวนสินค้าในคลังสินค้ามีไม่พอที่จะขายให้ลูกค้า เป็นต้น

จากปัญหาที่เกิดขึ้น จึงควรจัดทำระบบฐานข้อมูลสำหรับเก็บข้อมูลสินค้าในร้านขายยา เพื่อเพิ่มประสิทธิภาพในการจัดเก็บข้อมูลให้มีความเป็นระบบ ทำให้ง่ายในการค้นหาข้อมูลสินค้า เพื่อนำข้อมูลดังกล่าวไปใช้ประโยชน์ในการพัฒนาร้านให้ดีขึ้น เช่น การคำนวณราคาสินค้าและการค้นหาสินค้าภายในร้านได้รวดเร็วขึ้น และทำให้ทราบแนวโน้มการขายยาแต่ละประเภท ซึ่งจะง่ายในการคาดการณ์จำนวนสินค้าที่จะต้องสั่งซื้อเพื่อจัดเก็บในคลังสินค้าของร้าน เพื่อให้เพียงพอกับความต้องการของลูกค้า เป็นต้น

1.2 วัตถุประสงค์ของโครงการ

การพัฒนาระบบฐานข้อมูลร้านขายยา เพื่อช่วยการทำงานภายในร้าน มีวัตถุประสงค์ดังนี้

1. เพื่อศึกษาและออกแบบระบบฐานข้อมูลร้านขายยา
2. เพื่อสร้างระบบฐานข้อมูลร้านขายยา สำหรับจัดการข้อมูลการซื้อขาย และข้อมูลสินค้าในร้าน โดยใช้ซอฟต์แวร์ Oracle Database 11g
3. เพื่อบันทึกข้อมูลสินค้าและข้อมูลการซื้อขายลงในระบบฐานข้อมูลร้านขายยาโดยใช้คำสั่ง SQL
4. เพื่อค้นหาตำแหน่งของสินค้า ค้นหาสินค้าที่หมดในคลังสินค้า ค้นหาการขายสินค้าในแต่ละวัน ค้นหารายละเอียดการขายต่อ 1 ใบเสร็จรับเงิน และเรียกดูการจัดลำดับสินค้าที่ขายได้มากที่สุดไปหาน้อยสุดของร้านขายยาโดยใช้คำสั่ง SQL

1.3 ขอบเขตของโครงการ

ระบบฐานข้อมูลร้านขายยาถูกพัฒนาโดยใช้ซอฟต์แวร์ Oracle Database 11g มีขอบเขตในการทำงานดังนี้

1. สร้างระบบฐานข้อมูลเพื่อจัดเก็บข้อมูลการซื้อขายและข้อมูลสินค้าโดยใช้เครื่องมือ SQL Developer
2. สามารถเรียกข้อมูลสินค้าและข้อมูลการซื้อขายโดยใช้คำสั่ง SQL

1.4 ประโยชน์ที่ได้รับ

ระบบฐานข้อมูลร้านขายยาที่พัฒนา จะทำให้เกิดประโยชน์ดังนี้

1. มีระบบฐานข้อมูลสำหรับร้านขายยา ช่วยให้การจัดการข้อมูลในร้านมีประสิทธิภาพมากขึ้น
2. ช่วยลดความซ้ำซ้อนของการทำงานภายในร้าน ทำให้มีความสะดวกสบายในการขายสินค้า
3. มีความรู้ ความเข้าใจในระบบฐานข้อมูล เพื่อนำไปประยุกต์กับงานจริงได้

1.5 ขั้นตอนการดำเนินการ

ระบบฐานข้อมูลร้านขายยาที่พัฒนา มีขั้นตอนการดำนินงานดังนี้

1. ศึกษาลักษณะข้อมูลและผลลัพธ์ที่ต้องการของร้านขายยา
2. ศึกษาหลักการทำงานของซอฟต์แวร์ Oracle Database 11g
3. ศึกษาและออกแบบฐานข้อมูลร้านขายยา
4. สร้างฐานข้อมูลฐานข้อมูลร้านขายาโดยใช้เครื่องมือ SQL Developer
5. เพิ่มข้อมูลลงในฐานข้อมูลโดยใช้เครื่องมือ SQL Developer
6. เรียกดูข้อมูลในฐานข้อมูลร้านขายยาโดยใช้เครื่องมือ SQL Developer

บทที่ 2 ทฤษฎีที่เกี่ยวข้อง

ในบทนี้จะกล่าวถึงทฤษฎีที่เกี่ยวข้องกับโครงการนี้ เพื่อให้เข้าใจการออกแบบและการจัดการฐานข้อมูลสินค้าในร้านขายยา โดยมีสรุปเนื้อหาได้ดังนี้

2.1 ระบบฐานข้อมูล

ในการจัดเก็บรวบรวมข้อมูลที่มีความเกี่ยวข้องกันและมีความสัมพันธ์กัน โดยนำข้อมูลเหล่านี้ไปทำการสร้างระบบงานต่าง ๆ ให้เกิดประโยชน์มากที่สุด จำเป็นต้องมีฐานข้อมูลเพื่อจัดเก็บข้อมูลที่มีความเกี่ยวข้องสัมพันธ์กัน เพราะฐานข้อมูลมีระบบที่ช่วยในการจัดเก็บข้อมูลไม่ให้เกิดความซ้ำซ้อนของข้อมูล สามารถหลีกเลี่ยงความขัดแย้งของข้อมูลเกิดขึ้นในฐานข้อมูล มีการจัดเก็บข้อมูลอย่างเป็นระบบและเป็นมาตรฐานเดียวกัน อีกทั้งยังมีระบบรักษาความปลอดภัยในการเข้าถึงข้อมูล โดยการกำหนดสิทธิ์ผู้ใช้ในการเข้าถึงข้อมูล ซึ่งต้องมีซอฟต์แวร์ที่เป็นสื่อกลางในการติดต่อกันระหว่างผู้ใช้กับฐานข้อมูลเพื่อให้สามารถจัดการข้อมูลที่เก็บในฐานข้อมูลได้อย่างสะดวก และมีประสิทธิภาพมากขึ้น เรียกว่า ระบบจัดการฐานข้อมูล หรือ Database Management System (DBMS) [1]

2.2 แบบจำลองข้อมูล

ในการนำเสนอข้อมูลที่มีความเกี่ยวข้องสัมพันธ์กัน และมีโครงสร้างของระบบงานที่ซับซ้อน จำเป็นต้องมีเครื่องมือในการอธิบายให้อยู่ในรูปแบบที่บุคคลต่าง ๆ จะสามารถเข้าใจได้ง่าย และมีความเข้าใจตรงกัน จึงได้นำการใช้แผนภาพ อี-อาร์ไดอะแกรม (ER-Diagram) มาเสนอในการทำรายงานนี้ เพราะสามารถอธิบายความสัมพันธ์ของข้อมูลได้อย่างชัดเจน และเป็นแบบมาตรฐาน ส่วนแบบจำลองข้อมูลที่ใช้แพร่หลายมากที่สุด คือ แบบจำลองฐานข้อมูลเชิงสัมพันธ์ (Relational Database Model) เพราะนำเสนอความสัมพันธ์ของข้อมูลอยู่ในรูปแบบของตาราง ทำให้สามารถเข้าใจระบบที่ซับซ้อนได้เป็นอย่างดี [2]

2.3 แบบจำลองฐานข้อมูลเชิงสัมพันธ์

แบบจำลองฐานข้อมูลเชิงสัมพันธ์จะเก็บข้อมูลในรูปแบบตาราง 2 มิติ ที่ประกอบด้วยแถวและคอลัมน์ ซึ่งทำให้ง่ายต่อการเข้าใจและการนำไปประยุกต์ใช้งาน

ข้อมูลในหนึ่งตารางของแบบจำลองฐานข้อมูลเชิงสัมพันธ์สามารถเชื่อมโยงไปยังตารางอื่น ๆ ได้ด้วยความสัมพันธ์หลายแบบได้แก่ หนึ่งต่อหนึ่ง หนึ่งต่อกลุ่ม และ กลุ่มต่อกลุ่ม โดยที่มีคีย์บอกถึงความสัมพันธ์ระหว่างตารางด้วย ซึ่งคีย์ที่ใช้จะแบ่งเป็น 2 แบบ ดังนี้ [3]

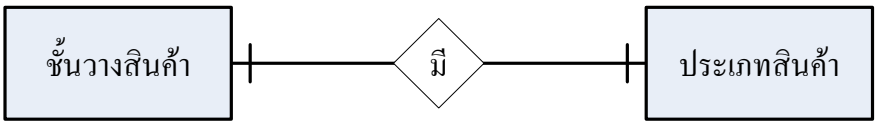
- คีย์หลัก (Primary Key:PK) คือคอลัมน์ที่ข้อมูลในแต่ละแถวไม่ซ้ำกัน โดยหนึ่งตารางควรมีคีย์หลักเพียง 1 คีย์ และคอลัมน์ที่เป็นคีย์หลักจะต้องไม่มีค่าว่าง (Null)
- คีย์นอก (Foreign Key:FK) คือคอลัมน์ของตารางหนึ่งที่มีความสัมพันธ์กับอีกคอลัมน์ของตารางอื่นที่เป็นคีย์หลัก โดยทั้งสองตารางสามารถเชื่อมโยงข้อมูลกันได้

2.4 องค์ประกอบการออกแบบฐานข้อมูลเชิงสัมพันธ์

เครื่องมือในการอธิบายความสัมพันธ์ระหว่างกลุ่มของข้อมูลในระบบงาน หรือความสัมพันธ์ระหว่างเอนทิตีกับเอนทิตี และใช้ในการออกแบบฐานข้อมูลเชิงสัมพันธ์ คือ แผนภาพอี-อาร์ไดอะแกรม (ER-Diagram) มีลักษณะเป็นแผนภาพ ทำให้สามารถมองภาพรวมของระบบงานได้ง่าย ซึ่งมีองค์ประกอบต่าง ๆ ที่สำคัญ ดังนี้ [2]

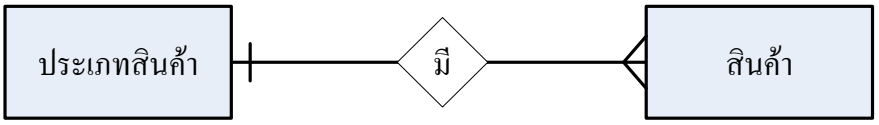
- เอนทิตี (Entity) คือ ข้อมูลที่บ่งบอกหรือบ่งชี้เอกลักษณ์เฉพาะตัวได้ ซึ่งเป็นข้อมูลที่มีความสัมพันธ์กันและต้องการรวบรวมลงในฐานข้อมูล เช่น บุคคล สิ่งของ สถานที่ หรือเหตุการณ์ต่าง ๆ โดยหลักเกณฑ์ในการตั้งชื่อเอนทิตีนั้นจะต้องเป็นคำนามเท่านั้น เช่น เอนทิตียา เอนทิตีใบสั่งซื้อยา เอนทิตีลูกค้า เอนทิตีพนักงาน เป็นต้น
- แอททริบิวต์ (Attribute) คือ ข้อมูลที่แสดงถึงรายละเอียด ส่วนประกอบ และคุณสมบัติของเอนทิตีนั้น ๆ เช่น เอนทิตียา ประกอบด้วยแอททริบิวต์ รหัสยา ชื่อยา ราคายา และจำนวนยาในคลังสินค้า เอนทิตีลูกค้า ประกอบด้วยแอททริบิวต์ รหัสลูกค้า ชื่อลูกค้า รหัสใบสั่งซื้อยา เป็นต้น
- ความสัมพันธ์ (Relationship) คือ เป็นการเชื่อมโยงความสัมพันธ์ระหว่างเอนทิตีกับเอนทิตีหรือมากกว่า 2 เอนทิตีขึ้นไปในระบบฐานข้อมูล โดยการเชื่อมความสัมพันธ์ระหว่างเอนทิตีนั้นจะใช้คำกริยาเป็นตัวเชื่อม ซึ่งจะขอยกตัวอย่างให้อยู่ในรูปแบบของ Crow’s Foot Model สามารถแบ่งความสัมพันธ์ออกเป็น 3 แบบ ได้แก่

แบบที่ 1 ความสัมพันธ์แบบหนึ่งต่อหนึ่ง (One to One Relationship) เป็นการแสดงความสัมพันธ์ของข้อมูลเอนทิตีหนึ่งว่า มีความสัมพันธ์ของหนึ่งข้อมูลกับอีกหนึ่งข้อมูลในอีกเอนทิตีหนึ่ง ซึ่งเป็นไปในลักษณะที่เป็นหนึ่งต่อหนึ่ง เช่น ชั้นวางสินค้า 1 ชั้นมีประเภทสินค้าได้เพียงประเภทเดียว และประเภทสินค้า 1 ประเภทตั้งอยู่บนชั้นวางสินค้าได้ 1 ชั้น



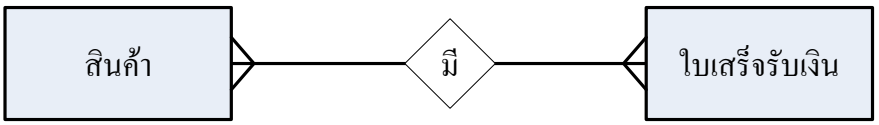
รูปที่ 2.1 ความสัมพันธ์แบบ หนึ่ง-ต่อ-หนึ่ง

แบบที่ 2 ความสัมพันธ์แบบหนึ่งต่อกลุ่ม (One to Many Relationship) เป็นการแสดงความสัมพันธ์ของข้อมูลของเอนทิตีหนึ่งว่า มีความสัมพันธ์ของหนึ่งข้อมูลกับอีกหลายข้อมูลในอีกเอนทิตีหนึ่ง เช่น ประเภทสินค้า 1 ประเภทมีสินค้าได้หลายชนิด และสินค้า 1 ชนิดมีประเภทสินค้าได้เพียงประเภทเดียวเท่านั้น



รูปที่ 2.2 ความสัมพันธ์แบบ หนึ่ง-ต่อ-กลุ่ม

แบบที่ 3 ความสัมพันธ์แบบกลุ่มต่อกลุ่ม (May to Many Relationship) เป็นการแสดงความสัมพันธ์ของข้อมูลของสองเอนทิตีในลักษณะแบบกลุ่มต่อกลุ่ม เช่น สินค้า 1 ชนิดอยู่ในใบเสร็จรับเงินได้หลายใบ และใบเสร็จรับเงิน 1 ใบมีสินค้าได้หลายชนิด



รูปที่ 2.3 ความสัมพันธ์แบบ กลุ่ม-ต่อ-กลุ่ม

2.5 ภาษาแอสคิวแอลในฐานข้อมูลออราเคิล

ในการจัดการและทำงานกับฐานข้อมูล นั้นต้องมีภาษากลางที่ใช้ในการติดต่อสื่อสารระหว่างผู้ใช้กับระบบฐานข้อมูลเชิงสัมพันธ์ เรียกว่า ภาษาเอสคิวแอล (SQL) ย่อมาจาก Structured Query Language เป็นภาษามาตรฐานในการเข้าถึง และจัดการฐานข้อมูล ทั้งการสร้างฐานข้อมูล เปลี่ยนแปลงข้อมูล และเรียกดูข้อมูลในฐานข้อมูล นอกจากนี้ระบบการจัดการฐานข้อมูลยังช่วยรักษาความปลอดภัยของข้อมูลโดยการกำหนดสิทธิ์ผู้ใช้ที่สามารถเข้ามาในฐานข้อมูลด้วย ซึ่งภาษา SQL จะสามารถแบ่งได้เป็น 4 กลุ่มคำสั่ง ดังนี้ [3]

1. Data Definition Language (DDL) เป็นคำสั่งที่ใช้สำหรับการสร้างและกำหนดโครงสร้างในฐานข้อมูล ได้แก่

- คำสั่ง CREATE ใช้สำหรับสร้างอ็อบเจกต์ลงในตารางของฐานข้อมูล มีรูปแบบดังนี้

```
CREATE TABLE [schema.] table_name
(column datatype [DEFAULT expr[, …]]);
```

- คำสั่ง ALTER ใช้สำหรับเปลี่ยนแปลงอ็อบเจกต์ในตารางของฐานข้อมูล
  - เพิ่มอ็อบเจกต์ในตารางของฐานข้อมูล มีรูปแบบดังนี้

```
ALTER TABLE table_name
ADD column_name datatype;
```

  - ลบอ็อบเจกต์ในตารางของฐานข้อมูล มีรูปแบบดังนี้

```
ALTER TABLE table_name
DROP COLUMN column_name;
```

  - แก้ไขอ็อบเจกต์ในตารางของฐานข้อมูล มีรูปแบบดังนี้

```
ALTER TABLE table_name
MODIFY column_name datatype;
```

- คำสั่ง DROP ใช้สำหรับลบอ็อบเจกต์ของตารางในฐานข้อมูล และเมื่อใช้คำสั่งนี้แล้วจะไม่สามารถใช้คำสั่ง Rollback ในการเรียกข้อมูลกลับมาได้ มีรูปแบบดังนี้

```
DROP TABLE table_name [PERGE];
```

- คำสั่ง TRUNCATE ใช้สำหรับลบข้อมูลทั้งตารางโดยไม่มีเงื่อนไข และเมื่อใช้คำสั่งนี้แล้วจะไม่สามารถใช้คำสั่ง Rollback ในการเรียกข้อมูลกลับมาได้ มีรูปแบบดังนี้

```
TRUNCATE TABLE table_name;
```

- คำสั่ง RENAME ใช้สำหรับการเปลี่ยนชื่อตารางในฐานข้อมูล มีรูปแบบดังนี้

```
ALTER TABLE table_name
RENAME column_name TO column_name;
```

2. Data Manipulation Language (DML) เป็นคำสั่งที่ใช้สำหรับการประมวลผลและจัดการกับข้อมูลในตาราง ไม่ว่าจะเป็นการเรียกดูข้อมูล เปลี่ยนแปลงข้อมูล และลบข้อมูลในตาราง ได้แก่

- คำสั่ง SELECT ใช้สำหรับเรียกดูข้อมูลในตาราง สามารถใส่เงื่อนไขในการเรียกดูได้ มีรูปแบบดังนี้

```
SELECT * | {[distinct] column_name | expression [alias], …}
FROM table_name
[WHERE condition]
[GROUP BY group_by_expression]
[HAVING group_condition]
[ORDER BY column_name];
```

- คำสั่ง INSERT ใช้สำหรับเพิ่มข้อมูลในตาราง มีรูปแบบดังนี้

```
INSERT INTO table_name [(column_name [, column_name…])]
VALUES (value [, value…]);
```

- คำสั่ง UPDATE ใช้สำหรับเปลี่ยนแปลงข้อมูลในตาราง มีรูปแบบดังนี้

```
UPDATE table_name
SET column_name = value [, column_name = value, …]
[WHERE condition];
```

- คำสั่ง DELETE ใช้สำหรับลบข้อมูลในตารางตามเงื่อนไขที่เรากำหนดได้ แต่ถ้าไม่ได้ทำการกำหนดเงื่อนไขจะเป็นการลบข้อมูลทั้งตาราง มีรูปแบบดังนี้

```
DELETE table_name
[WHERE condition];
```

- คำสั่ง MERGE ใช้สำหรับรวมข้อมูลระหว่าง 2 ตารางในฐานข้อมูล หรือเรียกอีกอย่างว่า Upsert โดยกรณีที่ไม่แน่ใจว่ามี Record ตามเงื่อนไขหรือไม่ จะแบ่งเป็น 2 กรณีคือ ถ้าไม่มี Record นี้จะทำการ INSERT Record แต่ถ้ามี Record นี้จะทำการ UPDATE Record แทน มีรูปแบบดังนี้

```
MERGE INTO table_name
USING table_name or query
ON (condition)
WHEN MATCH THEN update_clause
DELETE where_caluse
WHEN NOT MATCH THEN insert_clause;
```

3. Data Control Language (DCL) เป็นคำสั่งที่ใช้สำหรับการกำหนดสิทธิ์ของผู้ใช้ในการเข้าถึงฐานข้อมูล เพื่อควบคุมความปลอดภัยของฐานข้อมูล ได้แก่

- คำสั่ง GRANT ใช้สำหรับ กำหนดสิทธิ์ให้กับผู้ใช้ที่สามารถเข้าถึงฐานข้อมูล มีรูปแบบดังนี้

```
GRANT privileges ON table_name
TO grantees;
```

- คำสั่ง REVOKE ใช้สำหรับยกเลิกสิทธิ์ของผู้ใช้ในการเข้าถึงฐานข้อมูล มีรูปแบบดังนี้

```
REVOKE privileges ON table_name
FROM grantees;
```

4. Transaction Control Language (TCL) เป็นคำสั่งที่ใช้ควบคุมรายการเปลี่ยนแปลงในฐานข้อมูล ได้แก่

- คำสั่ง COMMIT ใช้สำหรับยืนยันคำสั่งการเปลี่ยนแปลงในฐานข้อมูล เพื่อให้ผลนั้นอยู่ตลอดไปจนกว่าจะมีคำสั่งมาเปลี่ยนแปลงข้อมูล
- คำสั่ง SAVEPOINT ใช้สำหรับกำหนดจุดเริ่มต้นของการเปลี่ยนแปลงในฐานข้อมูล
- คำสั่ง ROLLBACK ใช้สำหรับยกเลิกการเปลี่ยนแปลงของคำสั่งที่ส่งไปก่อนหน้า แต่คำสั่งนี้ไม่สามารถใช้กับคำสั่งจำพวก CREATE DROP และ ALTER ได้

บทที่ 3 ระเบียบวิธีการพัฒนา

การออกแบบระบบฐานข้อมูลเชิงสัมพันธ์สำหรับการจัดการซื้อขายสินค้าในร้านขายยา เพื่อใช้ในการจัดเก็บข้อมูลสินค้าให้มีความเป็นระบบ โดยมีวิธีการศึกษาและออกแบบระบบฐานข้อมูลดังนี้

การเก็บข้อมูลในร้านขายยาจะเป็นการจัดเก็บแบบแฟ้มข้อมูล ที่จะทำให้เกิดปัญหาต่อการจัดการสินค้าภายในร้าน ซึ่งส่งผลให้เกิดการคำนวณราคาสินค้ามีความล่าช้า การหาสินค้าภายในร้านไม่พบ และสินค้าในคลังสินค้าอาจมีจำนวนไม่พอต่อความต้องการของลูกค้า จากปัญหาดังกล่าวจึงควรจัดทำระบบฐานข้อมูลสำหรับเก็บข้อมูลสินค้าในร้านขายยา โดยจะเริ่มจากการศึกษาลักษณะของข้อมูลและผลลัพธ์ที่ต้องการของร้านขายยา แล้วนำข้อมูลที่ได้มาทำการออกแบบฐานข้อมูล ซึ่งจะแสดงดังในหัวข้อ 3.1 และ 3.2

3.1 ศึกษาลักษณะข้อมูลและผลลัพธ์ที่ต้องการของร้านขายยา

3.1.1 ลักษณะข้อมูลของร้านขายยา

ลักษณะข้อมูลของร้านขายยาจะเก็บในรูปแบบตาราง ซึ่งประกอบด้วยตารางดังนี้

- ตารางลูกค้า (CUSTOMER) ใช้สำหรับเก็บข้อมูลลูกค้าของร้านขายยา
- ตารางพนักงาน (EMPLOYEE) ใช้สำหรับเก็บข้อมูลพนักงานของร้านขายยา
- ตารางใบเสร็จรับเงิน (BILL) ใช้สำหรับเก็บข้อมูลใบเสร็จรับเงินแต่ละใบ
- ตารางรายละเอียดใบเสร็จรับเงิน (BILL_DETAIL) เก็บรายละเอียดแต่ละรายการในใบเสร็จรับเงิน
- ตารางสินค้า (PRODUCT) ใช้สำหรับเก็บข้อมูลสินค้าของร้านขายยา
- ตารางประเภทสินค้า (PRODUCT_TYPE) ใช้สำหรับเก็บข้อมูลประเภทสินค้าของร้านขายยา
- ตารางชั้นวางสินค้า (SHELF)ใช้สำหรับเก็บข้อมูลชั้นวางสินค้าของร้านขายยา

3.1.2 ผลลัพธ์ที่ต้องการของร้านขายยา

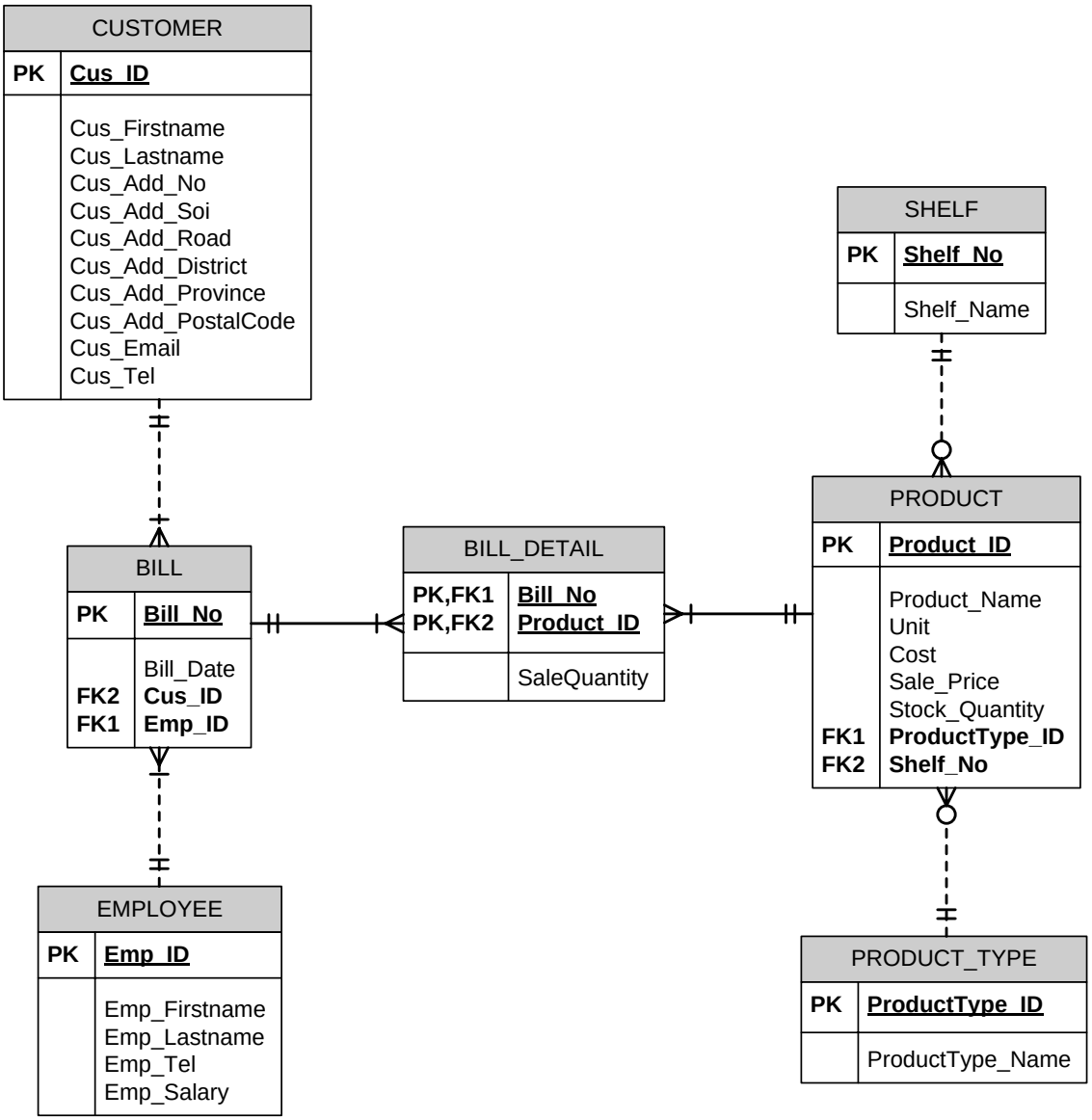
ผู้ใช้งานระบบที่เกี่ยวข้องกับฐานข้อมูลร้านขายยาคือ เจ้าของร้าน และพนักงานขาย ซึ่งผู้ใช้งานระบบมีความต้องการดังนี้

- สามารถเรียกดูตำแหน่งของสินค้า
- สามารถเรียกดูสินค้าที่หมดในคลังสินค้า
- สามารถเรียกดูยอดเงินรวมต่อ 1ใบเสร็จรับเงิน
- สามารถเรียกดูยอดเงินรวมของการขายสินค้าในแต่ละวัน
- สามารถเรียกดูสินค้าที่ขายในแต่ละวัน
- สามารถเรียกดูรายละเอียดการขายต่อ 1 ใบเสร็จรับเงิน
- สามารถเรียกดูลำดับสินค้าที่ขายได้มากที่สุดไปหาน้อยสุดได้

3.2 การออกแบบฐานข้อมูล

3.2.1 ความสัมพันธ์ระหว่างเอนทิตี (ER-Diagram)

การพัฒนาและออกแบบฐานข้อมูลของร้านขายยา ได้นำเสนอการใช้ฐานข้อมูลเชิงสัมพันธ์ ซึ่งแสดงอยู่ในรูปแบบแผนภาพ อี-อาร์ไดอะแกรม (ER-Diagram) เพื่ออธิบายความสัมพันธ์ระหว่างเอนทิตีหรือตารางของข้อมูล โดยการออกแบบฐานข้อมูลจะมีเอนทิตีที่เกี่ยวข้องแสดงดังในรูปที่ 3.1



รูปที่ 3.1 แบบจำลองความสัมพันธ์ของฐานข้อมูลร้านขายยา

ฐานข้อมูลร้านขายยา ประกอบด้วยเอนทิตีทั้งหมด 7 เอนทิตีดังในรูปที่ 3.1 โดยมีรายละเอียดของแต่ละเอนทิตีเป็นดังนี้

1. เอนทิตีลูกค้า (CUSTOMER) ประกอบด้วยข้อมูลรหัสลูกค้า ชื่อลูกค้า นามสกุลลูกค้า ที่อยู่ของลูกค้าซึ่งประกอบด้วย บ้านเลขที่ ซอย ถนน เขต จังหวัด รหัสไปรษณีย์ อีเมลของลูกค้า และเบอร์โทรศัพท์ของลูกค้า

2. เอนทิตีพนักงาน (EMPLOYEE) ประกอบด้วยข้อมูลรหัสพนักงาน ชื่อพนักงาน นามสกุลพนักงาน เบอร์โทรศัพท์ของพนักงาน และเงินเดือนพนักงาน

3. เอนทิตีใบเสร็จรับเงิน (BILL) ประกอบด้วยข้อมูลเลขที่ใบเสร็จรับเงิน วันที่ซื้อสินค้า รหัสลูกค้า และรหัสพนักงาน

4. เอนทิตีรายละเอียดใบเสร็จรับเงิน (BILL_DETAIL) ประกอบด้วยข้อมูลเลขที่ใบเสร็จรับเงิน รหัสสินค้า และจำนวนสินค้าที่ขาย

5. เอนทิตีสินค้า (PRODUCT) ประกอบด้วยข้อมูลรหัสสินค้า ชื่อสินค้า หน่วยของสินค้า ต้นทุนต่อหน่วย ราคาขายต่อหน่วย จำนวนสินค้าในคลังสินค้า รหัสประเภทสินค้า และหมายเลขชั้นวางสินค้า

6. เอนทิตีประเภทสินค้า (PRODUCT_TYPE) ประกอบด้วยข้อมูลรหัสประเภทสินค้า และชื่อประเภทสินค้า

7. เอนทิตีชั้นวางสินค้า (SHELF) ประกอบด้วยข้อมูลหมายเลขชั้นวางสินค้า และชื่อชั้นวางสินค้า

แบบจำลองความสัมพันธ์ของฐานข้อมูลร้านขายยาในรูปที่ 3.1 แสดงถึงความสัมพันธ์ระหว่างเอนทิตีในระบบ โดยจะมีรายละเอียดความสัมพันธ์ของแต่ละเอนทิตีเป็นดังนี้

- ความสัมพันธ์ระหว่างเอนทิตี BILL และเอนทิตี CUSTOMER เป็นแบบหนึ่งต่อกลุ่ม (1:M) โดยใบเสร็จรับเงินหนึ่งใบจะมีลูกค้าเพียงคนเดียวเท่านั้น และลูกค้าหนึ่งคนสามารถซื้อสินค้าได้หลาย ๆ ใบเสร็จ

- ความสัมพันธ์ระหว่างเอนทิตี BILL และเอนทิตี EMPLOYEE เป็นแบบหนึ่งต่อกลุ่ม (1:M) โดยใบเสร็จรับเงินหนึ่งใบจะมีพนักงานเพียงคนเดียวเท่านั้น และพนักงานหนึ่งคนสามารถขายสินค้าได้หลาย ๆ ใบเสร็จ

- ความสัมพันธ์ระหว่างเอนทิตี PRODUCT และเอนทิตี SHELF เป็นแบบหนึ่งต่อกลุ่ม (1:M) โดยสินค้าหนึ่งชนิดจะจัดวางบนชั้นเพียงหนึ่งชั้นเท่านั้น และชั้นวางของหนึ่งชั้นสามารถวางสินค้าได้หลายชนิดหรืออาจจะไม่มีวางเลยก็ได้ในกรณีที่ยังไม่ได้ระบุสินค้าที่จะวางขายบนชั้นนั้น

- ความสัมพันธ์ระหว่างเอนทิตี PRODUCT และเอนทิตี PRODUCT_TYPE เป็นแบบหนึ่งต่อกลุ่ม (1:M) โดยสินค้าหนึ่งชนิดสามารถเป็นได้แค่หนึ่งประเภทสินค้าเท่านั้น และประเภท

สินค้าหนึ่งประเภทประกอบด้วยสินค้าหลายชนิดหรืออาจจะไม่มีสินค้าอยู่ก็ได้ในกรณีสินค้ายังไม่ได้ระบุประเภทสินค้า

- ความสัมพันธ์ระหว่างเอนทิตี BILL_DETAIL และเอนทิตี BILL เป็นแบบหนึ่งต่อกลุ่ม (1:M) โดยรายละเอียดใบเสร็จรับเงินหนึ่งรายการสามารถเป็นได้แค่หนึ่งใบเสร็จรับเงินเท่านั้น และใบเสร็จ รับเงินหนึ่งใบประกอบด้วยรายละเอียดใบเสร็จรับเงินหลายรายการ

- ความสัมพันธ์ระหว่างเอนทิตี BILL_DETAIL และเอนทิตี PRODUCT เป็นแบบหนึ่งต่อกลุ่ม (1:M) โดยรายละเอียดใบเสร็จรับเงินหนึ่งรายการมีสินค้าเพียงชนิดเดียวเท่านั้น และสินค้าหนึ่งชนิดประกอบด้วยรายละเอียดใบเสร็จรับเงินหลายรายการ

3.2.2 โครงสร้างตารางข้อมูล

โครงสร้างของตารางจะบอกรายละเอียดเกี่ยวกับตารางข้อมูลต่าง ๆ ในฐานข้อมูล เพื่อทำให้เข้าใจข้อมูลในแต่ละตารางมากขึ้น รายละเอียดของโครงสร้างตารางข้อมูลมีแสดงดังตารางที่ 3.1 ถึงตารางที่ 3.7

ตารางที่ 3.1 ข้อมูลลูกค้า (CUSTOMER)

ชื่อฟิลด์	ชนิดข้อมูล (ขนาด)	ข้อบังคับ (constraints)	อ้างอิงจากตาราง	รายละเอียด
Cus_ID	NUMBER(5)	PK		รหัสลูกค้า
Cus_Firstname	VARCHAR2(50)			ชื่อลูกค้า
Cus_Lastname	VARCHAR2(50)			นามสกุลลูกค้า
Cus_Add_No	VARCHAR2(50)			บ้านเลขที่ที่ลูกค้าพักอาศัย
Cus_Add_Soi	VARCHAR2(50)			ซอยที่ลูกค้าพักอาศัย
Cus_Add_Road	VARCHAR2(50)			ถนนที่ลูกค้าพักอาศัย
Cus_Add_District	VARCHAR2(50)			เขตที่ลูกค้าพักอาศัย
Cus_Add_Province	VARCHAR2(50)			จังหวัดที่ลูกค้าพักอาศัย
Cus_Add_PostalCode	VARCHAR2(20)			รหัสไปรษณีย์ที่ลูกค้าพักอาศัย
Cus_Email	VARCHAR2(50)			อีเมลของลูกค้า
Cus_Tel	VARCHAR2 (10)			เบอร์โทรศัพท์ของลูกค้า

จากตารางที่ 3.1 แสดงรายละเอียดโครงสร้างตารางข้อมูลของลูกค้า สำหรับเก็บข้อมูลที่เกี่ยวกับลูกค้า โดยกำหนดให้ Cus_ID เป็นคีย์หลักของตาราง

ตารางที่ 3.2 ข้อมูลพนักงาน (EMPLOYEE)

ชื่อฟิลด์	ชนิดข้อมูล (ขนาด)	ข้อบังคับ (constraints)	อ้างอิงจากตาราง	รายละเอียด
Emp_ID	NUMBER(5)	PK		รหัสพนักงาน
Emp_Firstname	VARCHAR2(50)			ชื่อพนักงาน
Emp_Lastname	VARCHAR2(50)			นามสกุลพนักงาน
Emp_Tel	VARCHAR2(10)			เบอร์โทรศัพท์ของพนักงาน
Emp_Salary	NUMBER(10,2)			เงินเดือนของพนักงาน

จากตารางที่ 3.2 แสดงรายละเอียดโครงสร้างตารางข้อมูลของพนักงาน สำหรับเก็บข้อมูลที่เกี่ยวกับพนักงาน โดยกำหนดให้ Emp_ID เป็นคีย์หลักของตาราง

ตารางที่ 3.3 ข้อมูลใบเสร็จรับเงิน (BILL)

ชื่อฟิลด์	ชนิดข้อมูล (ขนาด)	ข้อบังคับ (constraints)	อ้างอิงจากตาราง	รายละเอียด
Bill_No	NUMBER(5)	PK		เลขที่ใบเสร็จรับเงิน
Bill_Date	DATE	DD-MON-YY		วันที่ซื้อสินค้า เช่น 01-FEB-14
Cus_ID	NUMBER(5)	FK	CUSTOMER	รหัสลูกค้า
Emp_ID	NUMBER(5)	FK	EMPLOYEE	รหัสพนักงาน

จากตารางที่ 3.3 แสดงรายละเอียดโครงสร้างตารางข้อมูลของใบเสร็จรับเงิน สำหรับเก็บข้อมูลที่เกี่ยวกับใบเสร็จรับเงิน โดยกำหนดให้ Bill_No เป็นคีย์หลักของตาราง ซึ่งมี Cus_ID เป็นคีย์นอกที่อ้างอิงไปยังตาราง CUSTOMER และมี Emp_ID เป็นคีย์นอกที่อ้างอิงไปยังตาราง EMPLOYEE

ตารางที่ 3.4 ข้อมูลรายละเอียดใบเสร็จรับเงิน (BILL_DETAIL)

ชื่อฟิลด์	ชนิดข้อมูล (ขนาด)	ข้อบังคับ (constraints)	อ้างอิงจากตาราง	รายละเอียด
Bill_No	NUMBER(5)	PK,FK	BILL	เลขที่ใบเสร็จรับเงิน
Product_ID	NUMBER(5)	PK,FK	PRODUCT	รหัสสินค้า
SaleQuantity	NUMBER(10)			จำนวนสินค้าที่ขาย

จากตารางที่ 3.4 แสดงรายละเอียดโครงสร้างตารางข้อมูลของรายละเอียดใบเสร็จรับเงิน สำหรับเก็บข้อมูลที่เกี่ยวกับรายละเอียดใบเสร็จรับเงิน โดยกำหนดให้ Bill_No และ Product_ID เป็นคีย์หลักร่วมกันของตาราง ซึ่งมี Bill_No เป็นคีย์นอกที่อ้างอิงไปยังตาราง BILL และมี Product_ID เป็นคีย์นอกที่อ้างอิงไปยังตาราง PRODUCT

ตารางที่ 3.5 ข้อมูลสินค้า (PRODUCT)

ชื่อฟิลด์	ชนิดข้อมูล (ขนาด)	ข้อบังคับ (constraints)	อ้างอิงจากตาราง	รายละเอียด
Product_ID	NUMBER(5)	PK		รหัสสินค้า
Product_Name	VARCHAR2(50)			ชื่อสินค้า
Unit	VARCHAR2 (20)			หน่วยของสินค้า
Cost	NUMBER(10,2)			ต้นทุนต่อหน่วย
Sale_Price	NUMBER(10,2)			ราคาขายต่อหน่วย
Stock_Quantity	NUMBER(10)			จำนวนสินค้าในคลังสินค้า

จากตารางที่ 3.5 แสดงรายละเอียดโครงสร้างตารางข้อมูลของสินค้า สำหรับเก็บข้อมูลที่เกี่ยวกับสินค้า โดยกำหนดให้ Product_ID เป็นคีย์หลักของตาราง ซึ่งมี ProductType_ID เป็นคีย์นอกที่อ้างอิงไปยังตาราง PRODUCT_TYPE และมี Shelf_No เป็นคีย์นอกที่อ้างอิงไปยังตาราง SHELF

ตารางที่ 3.6 ข้อมูลประเภทสินค้า (PRODUCT_TYPE)

ชื่อฟิลด์	ชนิดข้อมูล (ขนาด)	ข้อบังคับ (constraints)	อ้างอิงจากตาราง	รายละเอียด
ProductType_ID	NUMBER(4)	PK		รหัสประเภทสินค้า
ProductType_Name	VARCHAR2(50)			ชื่อประเภทสินค้า 4001 = Antibiotic 4002 = Specific diseases drug 4003 = Ophthalmic drugs 4004 = Gastrointestinal drugs 4005 = NSAIDs&muscle relaxant

จากตารางที่ 3.6 แสดงรายละเอียดโครงสร้างตารางข้อมูลของประเภทสินค้า สำหรับเก็บข้อมูลที่เกี่ยวกับประเภทสินค้า โดยกำหนดให้ ProductType_ID เป็นคีย์หลักของตาราง

ตารางที่ 3.7 ข้อมูลชั้นวางสินค้า (SHELF)

ชื่อฟิลด์	ชนิดข้อมูล (ขนาด)	ข้อบังคับ (constraints)	อ้างอิงจากตาราง	รายละเอียด
Shelf_No	NUMBER(4)	PK		หมายเลขชั้นวางสินค้า
Shelf_Name	VARCHAR2(50)			ชื่อชั้นวางสินค้า

จากตารางที่ 3.7 แสดงรายละเอียดโครงสร้างตารางข้อมูลของชั้นวางสินค้า สำหรับเก็บข้อมูลที่เกี่ยวกับชั้นวางสินค้า โดยกำหนดให้ Shelf_No เป็นคีย์หลักของตาราง

บทที่ 4 ผลการพัฒนา

เมื่อทำการออกแบบระบบฐานข้อมูลเสร็จเรียบร้อย จากนั้นจะนำฐานข้อมูลที่ได้ออกแบบมาทำการสร้างฐานข้อมูลจริงโดยใช้ซอฟต์แวร์ Oracle Database 11g แล้วใช้คำสั่งภาษาเอสคิวแอล (SQL) ในการจัดการกับฐานข้อมูล ไม่ว่าจะเป็นการสร้างตารางในฐานข้อมูล การเพิ่มข้อมูลลงในตาราง รวมถึงการเรียกดูข้อมูลที่อยู่ในฐานข้อมูล

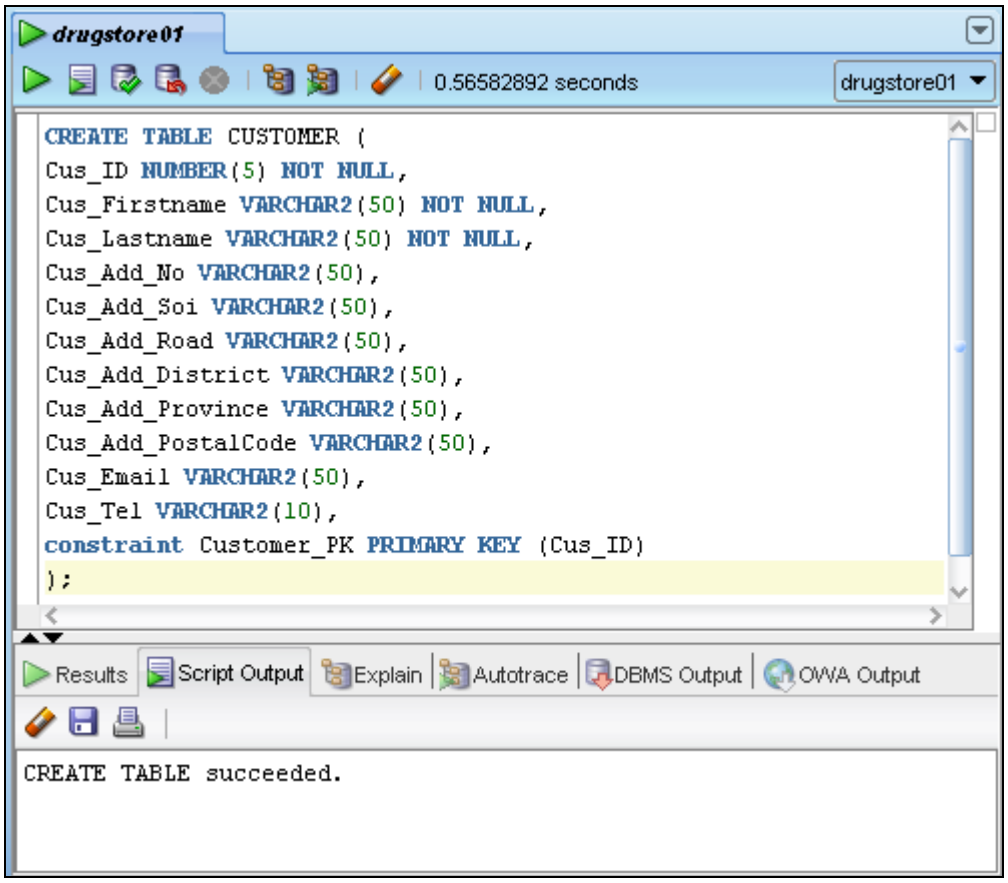
4.1 การสร้างตารางในฐานข้อมูล

การสร้างฐานข้อมูลจะเริ่มจากการสร้างตารางขึ้นมาก่อนเพื่อเก็บข้อมูล โดยใช้คำสั่ง CREATE เพื่อสร้างตารางในฐานข้อมูล

- ใช้คำสั่ง CREATE เพื่อสร้างตารางลูกค้า (CUSTOMER) ซึ่งมี SQL statement ดังนี้

```
CREATE TABLE CUSTOMER (
Cus_ID NUMBER(5) NOT NULL,
Cus_Firstname VARCHAR2(50) NOT NULL,
Cus_Lastname VARCHAR2(50) NOT NULL,
Cus_Add_No VARCHAR2(50),
Cus_Add_Soi VARCHAR2(50),
Cus_Add_Road VARCHAR2(50),
Cus_Add_District VARCHAR2(50),
Cus_Add_Province VARCHAR2(50),
Cus_Add_PostalCode VARCHAR2(50),
Cus_Email VARCHAR2(50),
Cus_Tel VARCHAR2(10),
constraint Customer_PK PRIMARY KEY (Cus_ID) );
```

- สร้างตารางลูกค้า (CUSTOMER) โดยใช้เครื่องมือ SQL Developer ดังรูปที่ 4.1



รูปที่ 4.1 สร้างตารางลูกค้า (CUSTOMER)

จากรูปที่ 4.1 เป็นการสร้างตารางลูกค้า (CUSTOMER) ซึ่งจะเก็บข้อมูลเกี่ยวกับลูกค้าดังนี้

1. รหัสลูกค้า (Cus_ID) กำหนดให้ชนิดของข้อมูลเป็นตัวเลข (NUMBER) ขนาด 5 หลัก และเป็นคีย์หลัก (Primary key) ซึ่งห้ามมีค่าใดค่าหนึ่งเป็นค่าว่าง (Not null) หรือลูกค้าทุกคนในตารางต้องมีรหัสลูกค้า
2. ชื่อลูกค้า (Cus_Firstname) กำหนดให้ชนิดของข้อมูลเป็นตัวอักษร (VARCHAR2) ขนาด 50 ตัวอักษร และห้ามมีค่าใดค่าหนึ่งเป็นค่าว่าง (Not null)
3. นามสกุลลูกค้า (Cus_Lastname) กำหนดให้ชนิดของข้อมูลเป็นตัวอักษร (VARCHAR2) ขนาด 50 ตัวอักษร และห้ามมีค่าใดค่าหนึ่งเป็นค่าว่าง (Not null)
4. บ้านเลขที่ที่ลูกค้าพักอาศัย (Cus_Add_No) กำหนดให้ชนิดของข้อมูลเป็นตัวอักษร (VARCHAR2) ขนาด 50 ตัวอักษร
5. ซอยที่ลูกค้าพักอาศัย (Cus_Add_Soi) กำหนดให้ชนิดของข้อมูลเป็นตัวอักษร (VARCHAR2) ขนาด 50 ตัวอักษร
6. ถนนที่ลูกค้าพักอาศัย (Cus_Add_Road) กำหนดให้ชนิดของข้อมูลเป็นตัวอักษร (VARCHAR2) ขนาด 50 ตัวอักษร

7. เขตที่ลูกค้าพักอาศัย (Cus_Add_District) กำหนดให้ชนิดของข้อมูลเป็นตัวอักษร (VARCHAR2) ขนาด 50 ตัวอักษร

8. จังหวัดที่ลูกค้าพักอาศัย (Cus_Add_Province) กำหนดให้ชนิดของข้อมูลเป็นตัวอักษร (VARCHAR2) ขนาด 50 ตัวอักษร

9. รหัสไปรษณีย์ที่ลูกค้าพักอาศัย (Cus_Add_PostalCode) กำหนดให้ชนิดของข้อมูลเป็นตัวอักษร (VARCHAR2) ขนาด 50 ตัวอักษร

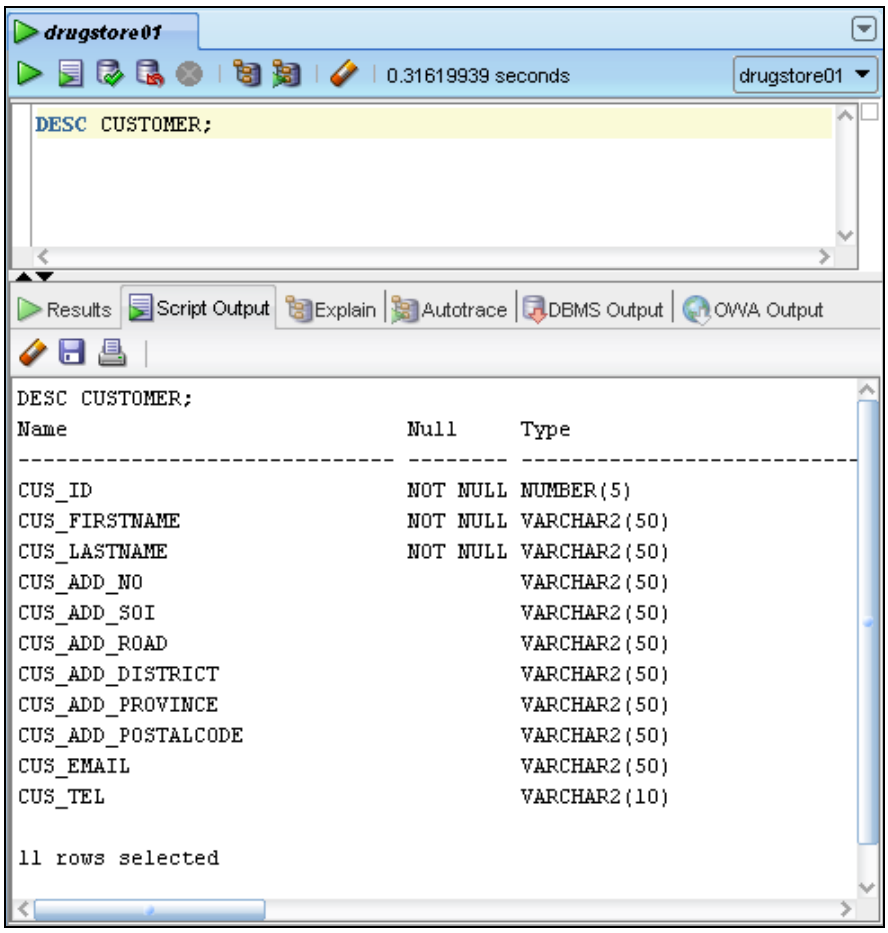
10. อีเมลลูกค้า (Cus_Email) กำหนดให้ชนิดของข้อมูลเป็นตัวอักษร (VARCHAR2) ขนาด 50 ตัวอักษร

11. เบอร์โทรศัพท์ของลูกค้า (Cus_Tel) กำหนดให้ชนิดของข้อมูลเป็นตัวอักษร (VARCHAR2) ขนาด 10 ตัวอักษร

- ใช้คำสั่ง DESC เพื่อเรียกดูโครงสร้างตารางลูกค้า (CUSTOMER) ซึ่งมี SQL statement ดังนี้

```
DESC CUSTOMER;
```

- ใช้คำสั่ง DESC เพื่อเรียกดูโครงสร้างตารางลูกค้า (CUSTOMER) โดยใช้เครื่องมือ SQL Developer ดังรูปที่ 4.2



รูปที่ 4.2 เรียกดูโครงสร้างตารางลูกค้า (CUSTOMER)

จากรูปที่ 4.2 เป็นการเรียกดูโครงสร้างตารางลูกค้า (CUSTOMER) จะประกอบด้วยทั้งหมด 6 คอลัมน์ ได้แก่ รหัสลูกค้า ชื่อลูกค้า นามสกุลลูกค้า ที่อยู่ลูกค้าซึ่งประกอบด้วยบ้านเลขที่ ซอย ถนน เขต จังหวัด รหัสไปรษณีย์ อีเมลลูกค้า และเบอร์โทศัพท์ลูกค้า ส่วนของตารางอื่น ๆ ในฐานข้อมูลร้านขายยาสามารถดูได้ที่ภาคผนวก ก

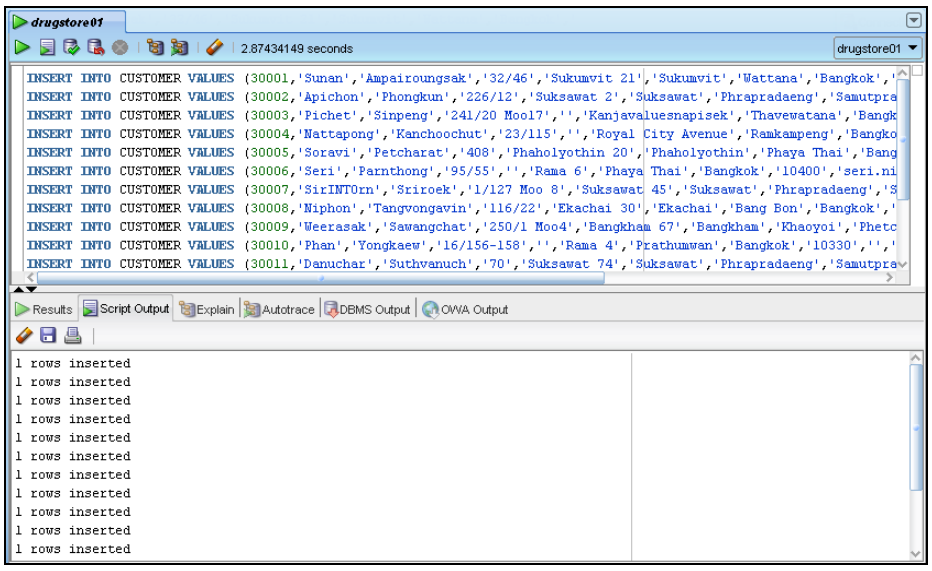
4.2 การเพิ่มข้อมูลในตาราง

เมื่อสร้างตารางในฐานข้อมูลเรียบร้อยแล้ว หลังจากนั้นจะทำการเพิ่มข้อมูลในตาราง โดยใช้คำสั่ง INSERT INTO

- ใช้คำสั่ง INSERT INTO เพื่อเพิ่มข้อมูลในตารางลูกค้า (CUSTOMER) ซึ่งมี SQL statement ดังนี้

```
INSERT INTO CUSTOMER VALUES (30001, 'Sunan', 'Ampairoungsak', '32/46',
'Sukumvit21', 'Sukumvit', 'Wattana', 'Bangkok', '10110',
'sunanamp96@hotmail.com', '0979098837');
INSERT INTO CUSTOMER VALUES (30002, 'Apichon', 'Phongkun', '226/12',
'Suksawat2', 'Suksawat', 'Phrapradaeng', 'Samutprakarn', '10130',
'tsnmach@hotmail.com', '0817201688');
INSERT INTO CUSTOMER VALUES (30003, 'Pichet', 'Sinpeng', '241/20
Moo17', '', 'Kanjavaluesnapisek', 'Thavewatana', 'Bangkok', '10170', '',
'0898852366');
```

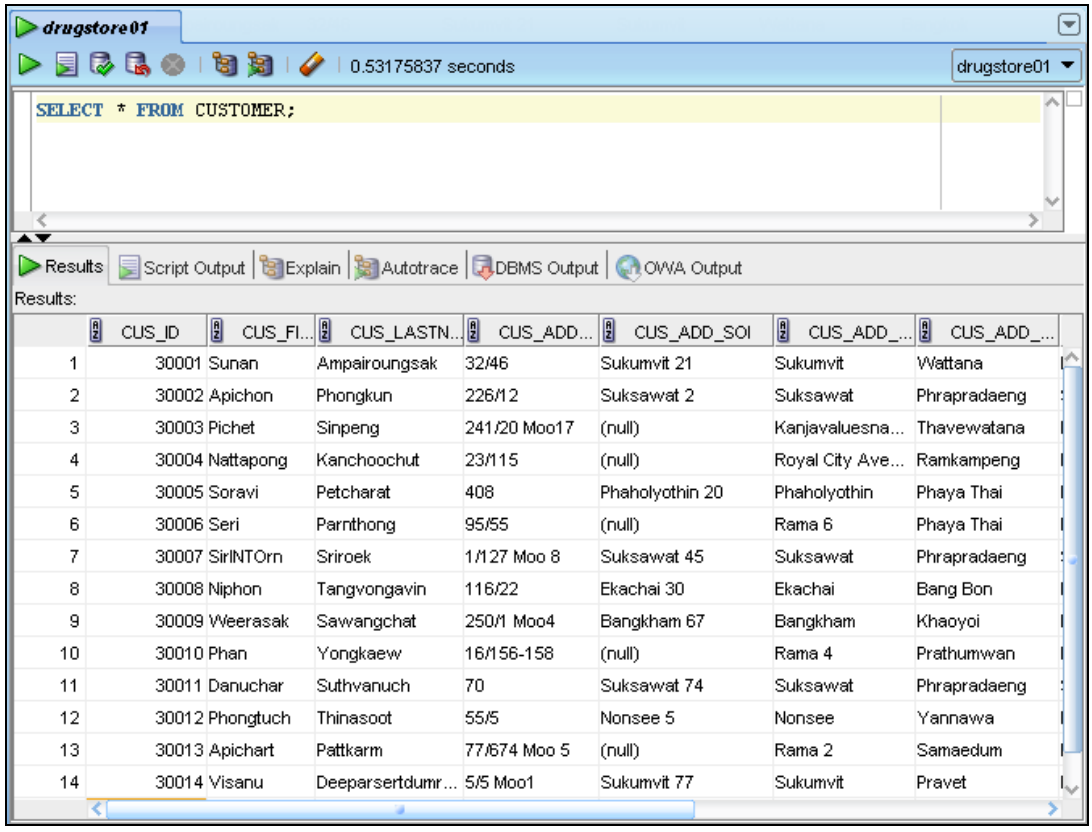
- ใช้คำสั่ง INSERT INTO เพื่อเพิ่มข้อมูลในตารางลูกค้า (CUSTOMER) โดยใช้เครื่องมือ SQL Developer ดังรูปที่ 4.3



รูปที่ 4.3 เพิ่มข้อมูลในตารางลูกค้า (CUSTOMER)

จากรูปที่ 4.3 เป็นการเพิ่มข้อมูลเข้าสู่ตารางลูกค้า (CUSTOMER) ซึ่งต้องใส่ข้อมูลดังนี้ รหัสลูกค้า (Cus_ID) ชื่อลูกค้า (Cus_Firstname) นามสกุลลูกค้า (Cus_Lastname) ที่อยู่ลูกค้าซึ่งประกอบด้วยบ้านเลขที่ (Cus_Add_No) ซอย (Cus_Add_Soi) ถนน (Cus_Add_Road) เขต (Cus_Add_District) จังหวัด (Cus_Add_Province) รหัสไปรษณีย์ (Cus_Add_PostalCode) อีเมลลูกค้า (Cus_Email) และเบอร์โทรศัพท์ลูกค้า (Cus_Tel)

หลังจากที่ใส่ข้อมูลลงในตารางเรียบร้อยแล้ว สามารถดูข้อมูลในตารางลูกค้า (CUSTOMER) ดังรูปที่ 4.4 ส่วนของการเพิ่มข้อมูลในตารางอื่น ๆ ของฐานข้อมูลร้านขายยาสามารถดูได้ที่ภาคผนวก ข



	CUS_ID	CUS_FI...	CUS_LASTN...	CUS_ADD...	CUS_ADD_SOI	CUS_ADD_...	CUS_ADD_...
1	30001	Sunan	Ampairoungsak	32/46	Sukumvit 21	Sukumvit	Wattana
2	30002	Apichon	Phongkun	226/12	Suksawat 2	Suksawat	Phrapradaeng
3	30003	Pichet	Sinpeng	241/20 Moo17	(null)	Kanjavaluesna...	Thavewatana
4	30004	Nattapong	Kanchoochut	23/115	(null)	Royal City Ave...	Ramkampeng
5	30005	Soravi	Petcharat	408	Phaholyothin 20	Phaholyothin	Phaya Thai
6	30006	Seri	Parnthong	95/55	(null)	Rama 6	Phaya Thai
7	30007	SirINTOrn	Sriroek	1/127 Moo 8	Suksawat 45	Suksawat	Phrapradaeng
8	30008	Niphon	Tangvongavin	116/22	Ekachai 30	Ekachai	Bang Bon
9	30009	Weerasak	Sawangchat	250/1 Moo4	Bangkham 67	Bangkham	Khaoyoi
10	30010	Phan	Yongkaew	16/156-158	(null)	Rama 4	Prathumwan
11	30011	Danuchar	Suthvanuch	70	Suksawat 74	Suksawat	Phrapradaeng
12	30012	Phongtuch	Thinasoot	55/5	Nonsee 5	Nonsee	Yannawa
13	30013	Apichart	Pattkarm	77/674 Moo 5	(null)	Rama 2	Samaedum
14	30014	Visanu	Deeparsertdumr...	5/5 Moo1	Sukumvit 77	Sukumvit	Pravet

รูปที่ 4.4 ข้อมูลที่เพิ่มในตารางลูกค้า (CUSTOMER)

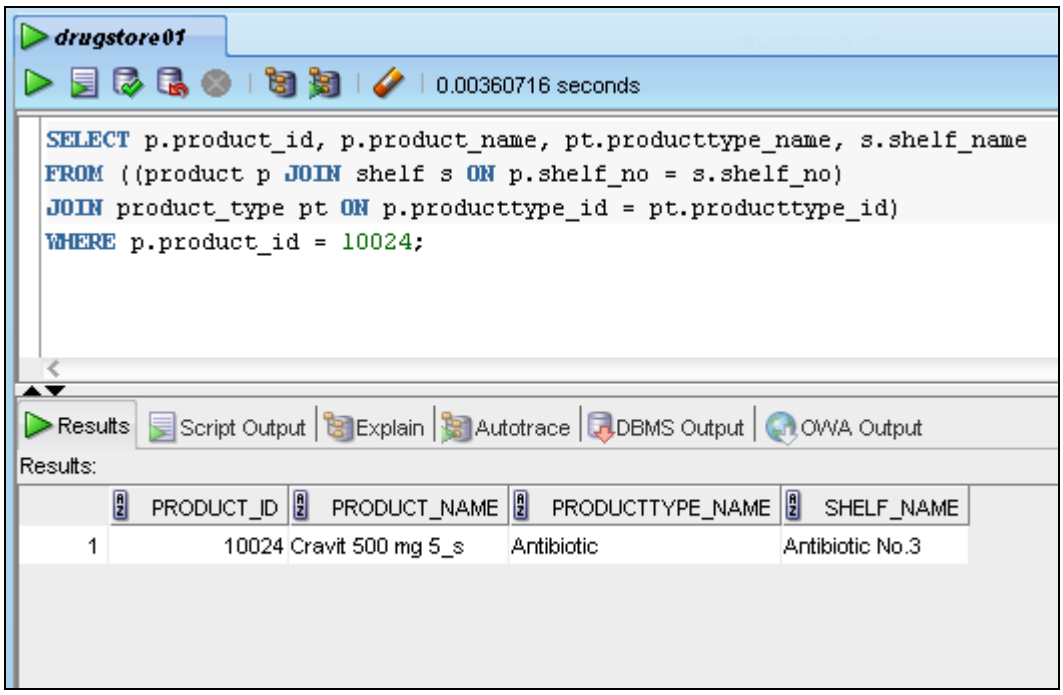
4.3 การเรียกดูข้อมูล

4.3.1 ค้นหาตำแหน่งของสินค้า

ในการค้นหาตำแหน่งของสินค้าจะพิจารณาสินค้าที่มีรหัสสินค้า (Product_ID) เป็น 10024 จะมี SQL statement ดังนี้

```
SELECT p.product_id, p.product_name, s.shelf_name
FROM product p JOIN shelf s ON p.shelf_no = s.shelf_no
WHERE p.product_id = 10024;
```

การค้นหาตำแหน่งของสินค้าที่มีรหัสสินค้า (Product_ID) เป็น 10024 โดยใช้เครื่องมือ SQL Developer ดังรูปที่ 4.5



รูปที่ 4.5 ค้นหาตำแหน่งของสินค้า

จากรูปที่ 4.5 แสดงการค้นหาตำแหน่งของสินค้า โดยจะแสดงผลบอกรหัสสินค้า ชื่อสินค้า ชื่อประเภทสินค้า และชื่อชั้นวางสินค้า โดยใช้คำสั่ง JOIN ON สำหรับดึงข้อมูลจาก 3 ตาราง ได้แก่

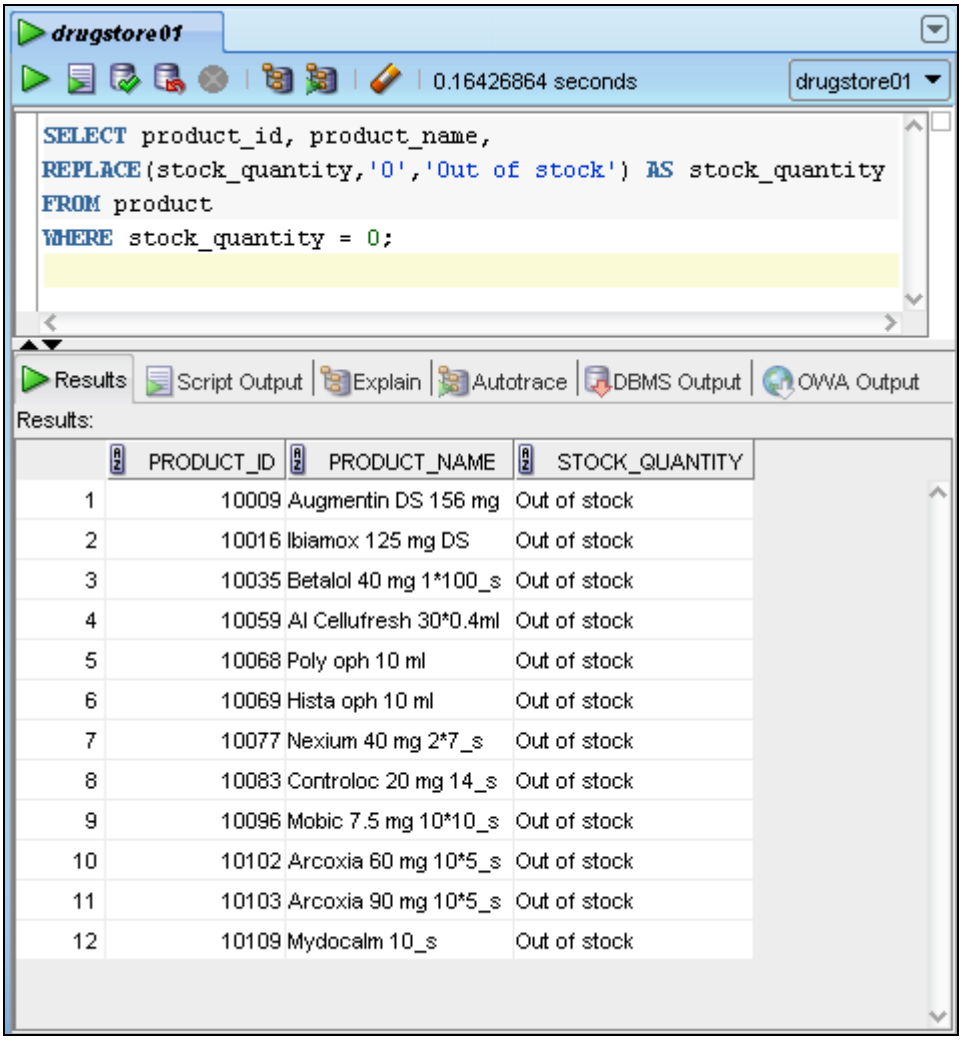
1. ตารางสินค้า (PRODUCT) เพื่อดึงรหัสสินค้า (Product_ID) และชื่อสินค้า (Product_Name) ที่เรากำหนด
2. ชั้นวางสินค้า (SHELF) เพื่อดึงชื่อชั้นวางสินค้า (Shelf_Name) ที่เราต้องการค้นหา
3. ตารางประเภทสินค้า (PRODUCT_TYPE) จะดึงชื่อประเภทสินค้า (ProductType_Name) ที่เราต้องการค้นหา

4.3.2 ค้นหาสินค้าที่หมดในคลังสินค้า

ในการค้นหาสินค้าที่หมดในคลังสินค้า จะมี SQL statement ดังนี้

```
SELECT product_id, product_name,
REPLACE(stock_quantity,'0','Out of stock') AS stock_quantity
FROM product
WHERE stock_quantity = 0;
```

การค้นหาสินค้าที่หมดในคลังสินค้าโดยใช้เครื่องมือ SQL Developer ดังรูปที่ 4.6



	PRODUCT_ID	PRODUCT_NAME	STOCK_QUANTITY
1	10009	Augmentin DS 156 mg	Out of stock
2	10016	Ibiamox 125 mg DS	Out of stock
3	10035	Betalol 40 mg 1*100_s	Out of stock
4	10059	Al Cellufresh 30*0.4ml	Out of stock
5	10068	Poly oph 10 ml	Out of stock
6	10069	Hista oph 10 ml	Out of stock
7	10077	Nexium 40 mg 2*7_s	Out of stock
8	10083	Controloc 20 mg 14_s	Out of stock
9	10096	Mobic 7.5 mg 10*10_s	Out of stock
10	10102	Arcoxia 60 mg 10*5_s	Out of stock
11	10103	Arcoxia 90 mg 10*5_s	Out of stock
12	10109	Mydocalm 10_s	Out of stock

รูปที่ 4.6 ค้นหาสินค้าที่หมดในคลังสินค้า

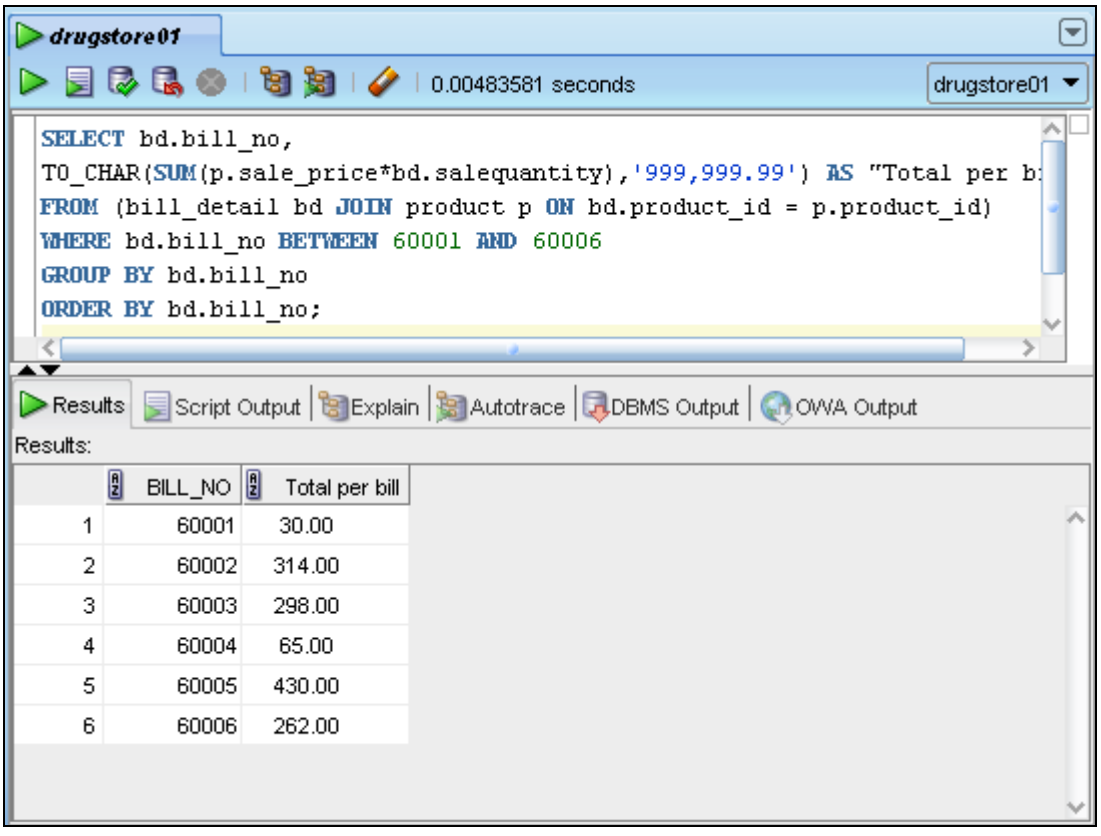
จากรูปที่ 4.6 แสดงการค้นหาสินค้าที่หมดในคลังสินค้า โดยจะแสดงผลบอกรหัสสินค้า ชื่อสินค้า และจำนวนสินค้าในคลังสินค้า ซึ่งสินค้าที่หมดสามารถหาได้จากจำนวนสินค้าในคลังสินค้า (Stock_Quantity) มีค่าเท่ากับ 0 ซึ่งจะแสดงที่หน้าจอว่า Out of stock

4.3.3 ค้นหายอดเงินรวมต่อ 1ใบเสร็จรับเงิน

ในการค้นหายอดเงินรวมต่อ 1 ใบเสร็จรับเงินจะพิจารณาเลขที่ใบเสร็จรับเงิน (Bill_No) เป็น 60001 ถึง 60006 จะมี SQL statement ดังนี้

```
SELECT bd.bill_no,
TO_CHAR(SUM(p.sale_price*bd.salequantity),'999,999.99') AS "Total per bill"
FROM (bill_detail bd JOIN product p ON bd.product_id = p.product_id)
WHERE bd.bill_no BETWEEN 60001 AND 60006
GROUP BY bd.bill_no
ORDER BY bd.bill_no;
```

การค้นหายอดเงินรวมต่อ 1 ใบเสร็จรับเงินที่มีเลขที่ใบเสร็จรับเงิน (Bill_No) เป็น 60001 ถึง 60006 โดยใช้เครื่องมือ SQL Developer ดังรูปที่ 4.7



รูปที่ 4.7 ค้นหายอดเงินรวมต่อ 1 ใบเสร็จรับเงิน

จากรูปที่ 4.7 แสดงการค้นหายอดเงินรวมต่อ 1 ใบเสร็จรับเงิน โดยจะแสดงผลบอกเลขที่ใบเสร็จรับเงิน และยอดเงินรวมในแต่ละใบเสร็จ ซึ่งใช้คำสั่งดังนี้

- คำสั่ง SUM สำหรับการคำนวณยอดเงินรวมในแต่ละใบเสร็จรับเงิน
- คำสั่ง BETWEEN…AND เพื่อกำหนดช่วงเลขที่ใบเสร็จรับเงิน
- คำสั่ง GROUP BY สำหรับการจัดกลุ่มตามเลขที่ใบเสร็จรับเงิน
- คำสั่ง ORDER BY สำหรับเรียงลำดับตามเลขที่ใบเสร็จรับเงิน
- คำสั่ง JOIN ON สำหรับดึงข้อมูลจาก 2 ตารางได้แก่

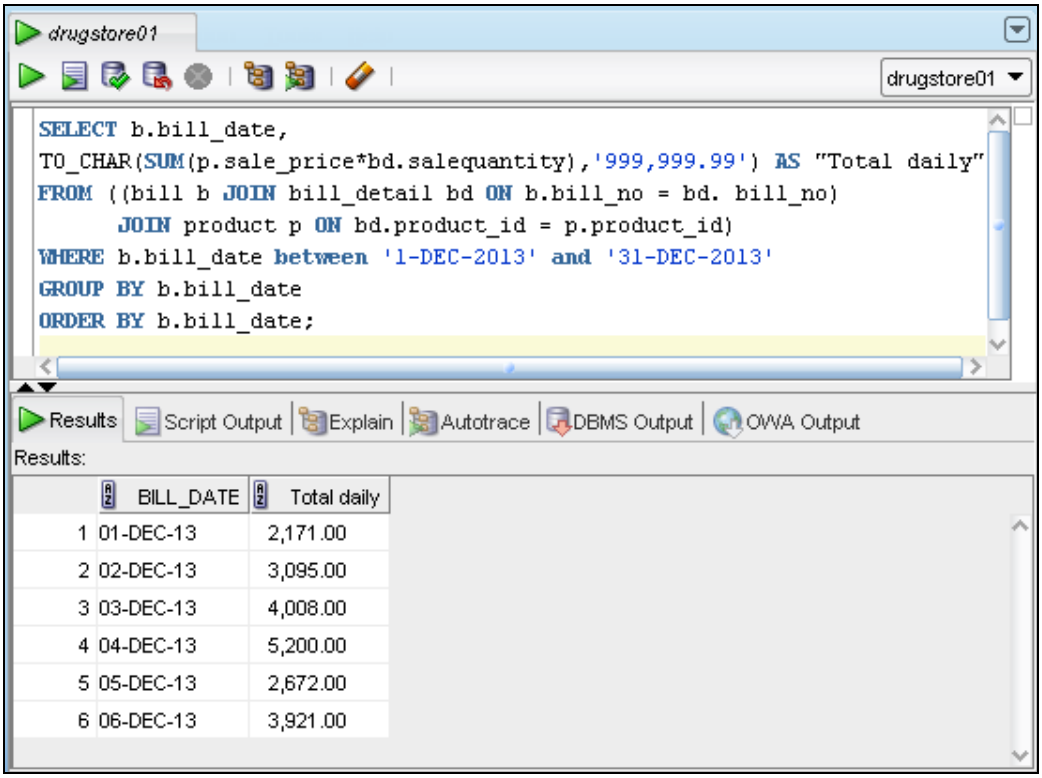
1. ตารางรายละเอียดใบเสร็จรับเงิน (BILL_DETAIL) เพื่อดึงจำนวนสินค้าที่ขาย (SaleQuantity) ของแต่ละใบเสร็จรับเงินที่กำหนด
2. ตารางสินค้า (PRODUCT) เพื่อดึงราคาขายต่อหน่วย (Sale_Price) ของสินค้าที่อยู่ในแต่ละใบเสร็จรับเงินที่กำหนด

4.3.4 ค้นหายอดเงินรวมของการขายสินค้าในแต่ละวัน

ในการค้นหายอดเงินรวมของการขายสินค้าในแต่ละวัน จะพิจารณาตั้งแต่วันที่ 1 ธันวาคม 2556 ถึง 31 ธันวาคม 2556 มี SQL statement ดังนี้

```
SELECT b.bill_date,
TO_CHAR(SUM(p.sale_price*bd.salequantity),'999,999.99') AS "Total daily"
FROM ((bill b JOIN bill_detail bd ON b.bill_no = bd. bill_no)
    JOIN product p ON bd.product_id = p.product_id)
WHERE b.bill_date between '1-DEC-2013' and '31-DEC-2013'
GROUP BY b.bill_date
ORDER BY b.bill_date;
```

การค้นหายอดเงินรวมของการขายสินค้าในแต่ละวันโดยใช้เครื่องมือ SQL Developer ดังรูปที่ 4.8



รูปที่ 4.8 ค้นหายอดเงินรวมของการขายสินค้าในแต่ละวัน

จากรูปที่ 4.8 แสดงการค้นหายอดเงินรวมของการขายสินค้าในแต่ละวัน โดยจะแสดงผลบอกวันที่ซื้อสินค้า และยอดเงินรวมในแต่ละวัน ซึ่งใช้คำสั่งดังนี้

- คำสั่ง SUM สำหรับการคำนวณยอดเงินรวมในแต่ละวัน
- คำสั่ง BETWEEN…AND เพื่อกำหนดช่วงวันที่ซื้อสินค้า
- คำสั่ง GROUP BY สำหรับการจัดกลุ่มตามวันที่ซื้อสินค้า
- คำสั่ง ORDER BY สำหรับเรียงลำดับตามวันที่ซื้อสินค้า
- คำสั่ง JOIN ON เพื่อดึงข้อมูลจาก 3 ตารางได้แก่

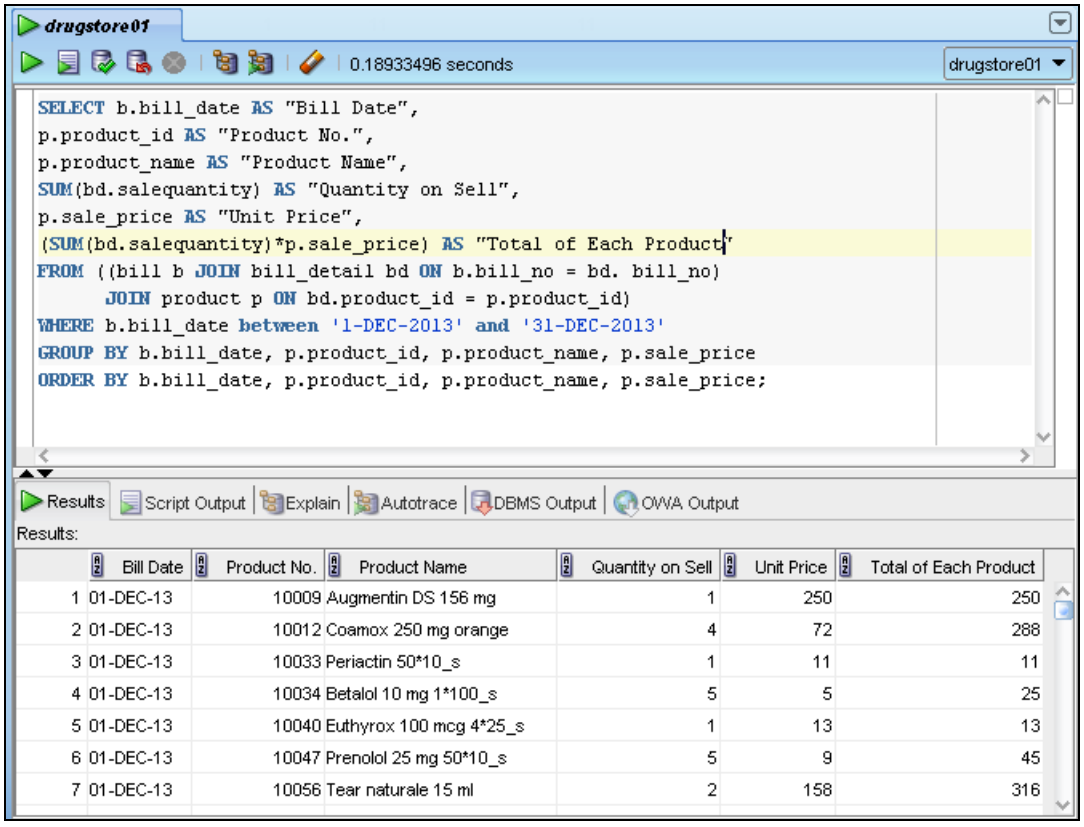
1. ตารางใบเสร็จรับเงิน (BILL) เพื่อดึงวันที่ซื้อสินค้า (Bill_Date) ที่กำหนด
2. ตารางรายละเอียดใบเสร็จรับเงิน (BILL_DETAIL) เพื่อดึงจำนวนสินค้าที่ขาย (SaleQuantity) ของแต่ละวันที่กำหนด
3. ตารางสินค้า (PRODUCT) เพื่อดึงราคาขายต่อหน่วย (Sale_Price) ของสินค้าที่อยู่ในแต่ละวันที่กำหนด

4.3.5 ค้นหาสินค้าที่ขายในแต่ละวัน

ในการค้นหาสินค้าที่ขายในแต่ละวัน จะพิจารณาตั้งแต่วันที่ 1 ธันวาคม 2556 ถึง 31 ธันวาคม 2556 มี SQL statement ดังนี้

```
SELECT b.bill_date AS "Bill Date",
p.product_id AS "Product No.",
p.product_name AS "Product Name",
SUM(bd.salequantity) AS "Quantity on Sell",
p.sale_price AS "Unit Price",
(SUM(bd.salequantity)*p.sale_price) AS "Total of Each Product"
FROM ((bill b JOIN bill_detail bd ON b.bill_no = bd. bill_no)
    JOIN product p ON bd.product_id = p.product_id)
WHERE b.bill_date between '1-DEC-2013' and '31-DEC-2013'
GROUP BY b.bill_date, p.product_id, p.product_name, p.sale_price
ORDER BY b.bill_date, p.product_id, p.product_name, p.sale_price;
```

การค้นหาสินค้าที่ขายในแต่ละวันโดยใช้เครื่องมือ SQL Developer ดังรูปที่ 4.9



	Bill Date	Product No.	Product Name	Quantity on Sell	Unit Price	Total of Each Product
1	01-DEC-13	10009	Augmentin DS 156 mg	1	250	250
2	01-DEC-13	10012	Coamox 250 mg orange	4	72	288
3	01-DEC-13	10033	Periactin 50*10_s	1	11	11
4	01-DEC-13	10034	Betalol 10 mg 1*100_s	5	5	25
5	01-DEC-13	10040	Euthyrox 100 mcg 4*25_s	1	13	13
6	01-DEC-13	10047	Prenolol 25 mg 50*10_s	5	9	45
7	01-DEC-13	10056	Tear naturale 15 ml	2	158	316

รูปที่ 4.9 ค้นหาสินค้าที่ขายในแต่ละวัน

จากรูปที่ 4.9 แสดงการค้นหาสินค้าที่ขายในแต่ละวัน โดยจะแสดงผลบอกวันที่ซื้อสินค้า รหัสสินค้า ชื่อสินค้า จำนวนสินค้าที่ขาย ราคาต่อหน่วย และยอดเงินรวมของสินค้าแต่ละชนิด ซึ่งใช้คำสั่งดังนี้

- คำสั่ง SUM สำหรับการคำนวณผลรวมจำนวนสินค้าที่ขายในแต่ละวัน
- คำสั่ง BETWEEN…AND เพื่อกำหนดช่วงวันที่ซื้อสินค้า
- คำสั่ง GROUP BY สำหรับการจัดกลุ่มตามวันที่ซื้อสินค้า รหัสสินค้า ชื่อสินค้า จำนวนสินค้าที่ขาย และราคาต่อหน่วย
- คำสั่ง ORDER BY สำหรับเรียงลำดับตามวันที่ซื้อสินค้า รหัสสินค้า ชื่อสินค้า จำนวนสินค้าที่ขาย และราคาต่อหน่วย
- คำสั่ง JOIN ON เพื่อดึงข้อมูลจาก 3 ตารางได้แก่

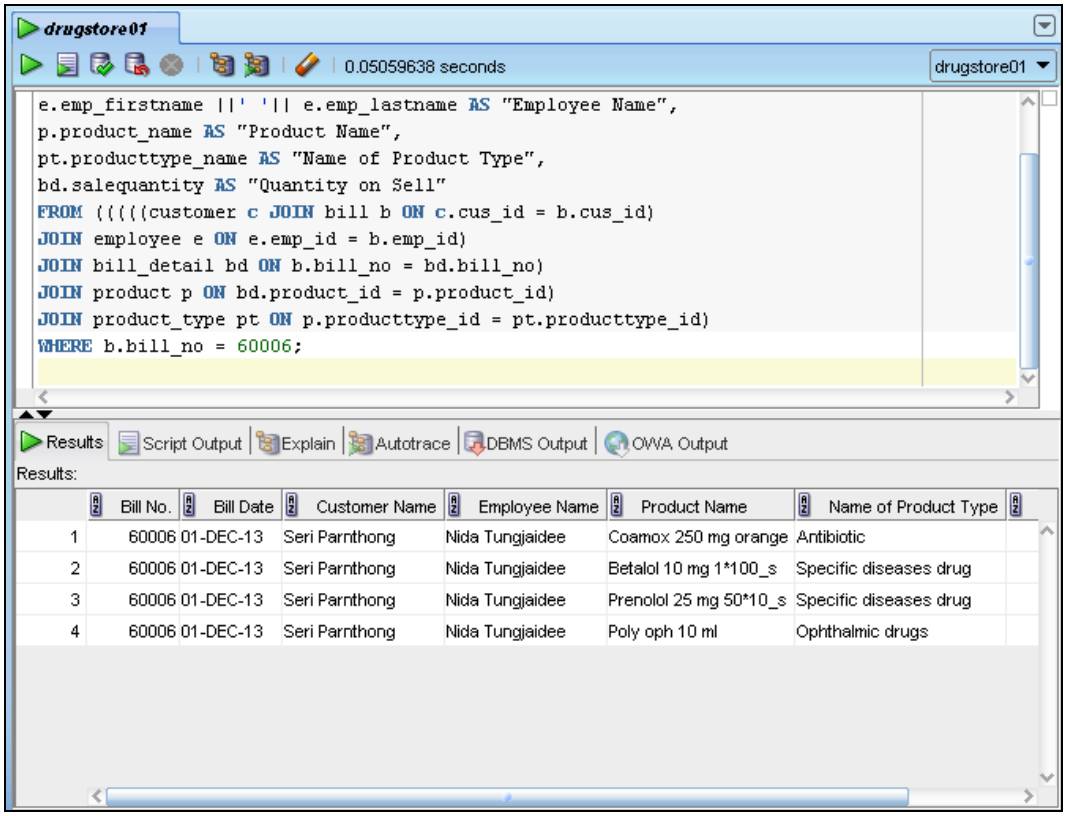
1. ตารางใบเสร็จรับเงิน (BILL) เพื่อดึงวันที่ซื้อสินค้า (Bill_Date) ที่กำหนด
2. ตารางรายละเอียดใบเสร็จรับเงิน (BILL_DETAIL) เพื่อดึงจำนวนสินค้าที่ขาย (SaleQuantity) ของแต่ละวันที่กำหนด
3. ตารางสินค้า (PRODUCT) เพื่อดึงชื่อสินค้า (Product_Name) ในแต่ละวันที่กำหนด

4.3.6 ค้นหารายละเอียดการขายต่อ 1 ใบเสร็จรับเงิน

ในการค้นหารายละเอียดการขายต่อ 1 ใบเสร็จรับเงิน จะพิจารณาการค้นหาของเลขที่ใบเสร็จรับเงิน 60006 มี SQL statement ดังนี้

```
SELECT b.bill_no AS "Bill No.",b.bill_date AS "Bill Date",
c.cus_firstname ||' '|| c.cus_lastname AS "Customer Name",
e.emp_firstname ||' '|| e.emp_lastname AS "Employee Name",
p.product_name AS "Product Name",
pt.producttype_name AS "Name of Product Type",
bd.salequantity AS "Quantity on Sell"
FROM (((((customer c JOIN bill b ON c.cus_id = b.cus_id)
JOIN employee e ON e.emp_id = b.emp_id)
JOIN bill_detail bd ON b.bill_no = bd.bill_no)
JOIN product p ON bd.product_id = p.product_id)
JOIN product_type pt ON p.producttype_id = pt.producttype_id)
WHERE b.bill_no = 60006;
```

การค้นหารายละเอียดการขายต่อ 1 ใบเสร็จรับเงินโดยใช้เครื่องมือ SQL Developer ดังรูปที่ 4.10



รูปที่ 4.10 ค้นหารายละเอียดการขายต่อ 1 ใบเสร็จรับเงิน

จากรูปที่ 4.10 แสดงการค้นหารายละเอียดการขายต่อ 1 ใบเสร็จรับเงิน เราจะพิจารณาเลขที่ใบเสร็จรับเงิน 60006 โดยจะแสดงผลบอกเลขที่ใบเสร็จรับเงิน วันที่ซื้อสินค้า ชื่อลูกค้า นามสกุลลูกค้า ชื่อพนักงาน นามสกุลพนักงาน ชื่อสินค้า ชื่อประเภทสินค้า และจำนวนสินค้าที่ขาย โดยใช้คำสั่ง JOIN ON เพื่อดึงข้อมูลจาก 6 ตาราง ได้แก่

1. ตารางลูกค้า (CUSTOMER) เพื่อดึงชื่อและนามสกุลลูกค้าที่ทำการซื้อสินค้าในใบเสร็จรับเงินที่กำหนด

2. ตารางพนักงาน (EMPLOYEE) เพื่อดึงชื่อและนามสกุลพนักงานที่ทำการขายสินค้าในใบเสร็จรับเงินที่กำหนด

3. ตารางใบเสร็จรับเงิน (BILL) เพื่อดึงวันที่ซื้อสินค้า (Bill_Date) ของใบเสร็จรับเงินที่กำหนด

4. ตารางรายละเอียดใบเสร็จรับเงิน (BILL_DETAIL) เพื่อดึงจำนวนสินค้าที่ขาย (SaleQuantity) ได้ในใบเสร็จรับเงินที่กำหนด

5. ตารางสินค้า (PRODUCT) เพื่อดึงชื่อสินค้า (Product_Name) ที่ขายได้ในใบเสร็จรับเงินที่กำหนด

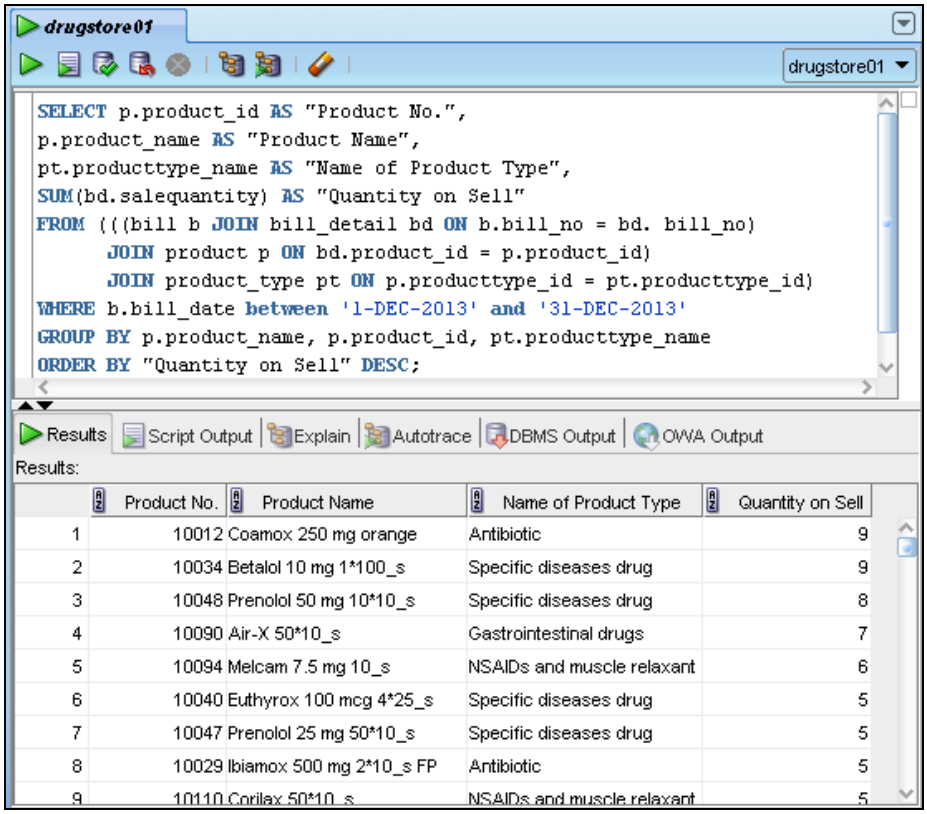
6. ตารางประเภทสินค้า (PRODUCT_TYPE) เพื่อดึงชื่อประเภทสินค้า (ProductType_Name) ที่ขายได้ของใบเสร็จรับเงินที่กำหนด

4.3.7 ค้นหาลำดับสินค้าที่ขายได้มากที่สุดไปหาน้อยสุด

ในการค้นหาลำดับสินค้าที่ขายได้มากที่สุดไปหาน้อยสุด จะพิจารณาตั้งแต่วันที่ 1 ธันวาคม 2556 ถึง 31 ธันวาคม 2556 มี SQL statement ดังนี้

```
SELECT p.product_id AS "Product No.",
p.product_name AS "Product Name",
pt.producttype_name AS "Name of Product Type",
SUM(bd.salequantity) AS "Quantity on Sell"
FROM (((bill b JOIN bill_detail bd ON b.bill_no = bd. bill_no)
    JOIN product p ON bd.product_id = p.product_id)
    JOIN product_type pt ON p.producttype_id = pt.producttype_id)
WHERE b.bill_date between '1-DEC-2013' and '31-DEC-2013'
GROUP BY p.product_name, p.product_id, pt.producttype_name
ORDER BY "Quantity on Sell" DESC;
```

การค้นหาลำดับสินค้าที่ขายได้มากที่สุดไปหาน้อยสุด สามารถหาได้จากจำนวนสินค้าที่ขายได้ โดยใช้เครื่องมือ SQL Developer ดังรูปที่ 4.11



รูปที่ 4.11 ค้นหาลำดับสินค้าที่ขายได้มากที่สุดไปหาน้อยสุด

จากรูปที่ 4.11 แสดงการค้นหาลำดับสินค้าที่ขายได้มากที่สุดไปหาน้อยสุด โดยนำจำนวนสินค้าที่ขายได้ในแต่ละชนิดมาเรียงจากมากไปหาน้อย จะแสดงผลบอกรหัสสินค้า ชื่อสินค้า ชื่อประเภทสินค้า จำนวนสินค้าที่ขาย ซึ่งใช้คำสั่งดังนี้

- คำสั่ง SUM สำหรับการคำนวณผลรวมจำนวนสินค้าที่ขายดีประจำเดือนที่กำหนด
- คำสั่ง BETWEEN…AND เพื่อกำหนดช่วงวันที่ของการจัดอันดับสินค้าขายดี
- คำสั่ง GROUP BY สำหรับการจัดกลุ่มตามชื่อสินค้า รหัสสินค้า และชื่อประเภทสินค้า
- คำสั่ง ORDER BY สำหรับเรียงลำดับตามจำนวนสินค้าที่ขายได้ในแต่ละชนิด
- คำสั่ง JOIN ON เพื่อดึงข้อมูลจาก 3 ตารางได้แก่

1. ตารางใบเสร็จรับเงิน (BILL) เพื่อดึงวันที่ซื้อสินค้า (Bill_Date) ในระหว่างที่ทำการจัดอันดับสินค้าขายดี
2. ตารางรายละเอียดใบเสร็จรับเงิน (BILL_DETAIL) เพื่อดึงจำนวนสินค้าที่ขาย (SaleQuantity) ของเดือนที่จัดอันดับขายดี
3. ตารางสินค้า (PRODUCT) เพื่อดึงชื่อสินค้า (Product_Name) ที่ขายดีประจำเดือน
4. ตารางประเภทสินค้า (PRODUCT_TYPE) เพื่อดึงชื่อประเภทสินค้า (ProductType_Name)

บทที่ 5 อภิปรายผล

จากโครงงานนี้ได้ทำการออกแบบและพัฒนาระบบฐานข้อมูล สามารถสรุปผลการพัฒนา และมีแนวทางการพัฒนาเพื่อปรับปรุงระบบฐานข้อมูลให้ทำงานได้มีประสิทธิภาพมากขึ้นได้ดังนี้

5.1 สรุปผลการพัฒนา

การออกแบบและพัฒนาระบบฐานข้อมูลร้านขายยา เพื่อจัดเก็บข้อมูลการซื้อขายและข้อมูลสินค้าลงในฐานข้อมูล มีการดำเนินการโดยสรุปดังนี้

1. ทำการออกแบบฐานข้อมูลโดยแสดงในรูปแบบแผนภาพอี-อาร์ไดอะแกรม (ER-Diagram) เพื่ออธิบายความสัมพันธ์ระหว่างเอนทิตี

2. ทำการสร้างระบบฐานข้อมูลร้านขายยาให้สอดคล้องกับการออกแบบในแผนภาพอี-อาร์ไดอะแกรม (ER-Diagram) ซึ่งจะมีตารางที่สร้างขึ้นได้แก่ ตารางลูกค้า ตารางพนักงาน ตารางใบเสร็จรับเงิน ตารางรายละเอียดใบเสร็จรับเงิน ตารางสินค้า ตารางประเภทสินค้า และตารางชั้นวางสินค้า

3. ทำการเพิ่มข้อมูลสินค้าและข้อมูลการซื้อขายลงในฐานข้อมูลร้านขายยา

4. ทำการเรียกดูข้อมูลสินค้าและข้อมูลการซื้อขายในฐานข้อมูล ซึ่งสามารถค้นหาตำแหน่งของสินค้า ค้นหาสินค้าที่หมดในคลังสินค้า ค้นหายอดเงินรวมต่อ 1 ใบเสร็จรับเงิน ค้นหายอดเงินรวมของการขายสินค้าในแต่ละวัน ค้นหาสินค้าที่ขายในแต่ละวัน รายงานรายละเอียดการขายต่อ 1 ใบเสร็จรับเงิน และค้นหาการจัดลำดับสินค้าที่ขายได้มากที่สุดไปหาน้อยสุด

จากการพัฒนาระบบฐานข้อมูลร้านขายยาที่จะเก็บข้อมูลลงในฐานข้อมูล จะทำให้มีการจัดเก็บข้อมูลอย่างเป็นระบบ ซึ่งส่งผลให้มีการค้นหาข้อมูลต่าง ๆ ได้รวดเร็วขึ้น และยังทำให้แนวโน้มการขายยาแต่ละประเภท ซึ่งจะง่ายในการคาดการณ์จำนวนสินค้าที่จะต้องสั่งซื้อเพื่อจัดเก็บในคลังสินค้าของร้าน เพื่อให้เพียงพอกับความต้องการของลูกค้า โดยผู้จัดทำได้สร้างฐานข้อมูลด้วย Oracle 11g โดยใช้คำสั่ง SQL Developer ตามแผนที่ได้วางไว้ และได้ผลออกมาสำเร็จ แต่มีข้อจำกัดในเรื่องเวลาที่ใช้จัดทำเล่มรายงาน ทำให้ใส่ข้อมูลลงในฐานข้อมูลได้ไม่มากนัก แต่ใส่ข้อมูลเข้าไปจำนวนหนึ่งเพื่อให้สามารถทำการทดลอง เรียกดูข้อมูล และออกรายงานได้

5.2 ข้อเสนอแนะ

ระบบฐานข้อมูลที่ใช้สำหรับเก็บข้อมูลภายในร้านขายยา เป็นระบบที่พัฒนาขึ้นเพื่อจัดเก็บยอดขายของร้านเท่านั้น ยังไม่รวมถึงการจัดการด้านอื่น ๆ เช่น ระบบการให้สิทธิ์กับพนักงานในร้าน การจัดการด้านรายจ่าย ซึ่งจะนำไปต่อยอดเพื่อให้ระบบมีประสิทธิภาพและสมบรูณ์มากยิ่งขึ้น และในการเพิ่มความสะดวกในการใช้งานนั้น ควรจะเพิ่มส่วนของ Graphic User Interface เพื่อช่วยให้สะดวกและมีความสวยงามมากยิ่งขึ้น

เอกสารอ้างอิง

[1] Billings, M. and et al., 2008, “Oracle Database”, In **Oracle Database 11g: Administration Workshop I**, Vol.1, Edition 1.1, Oracle Corporation.

[2] Singh, P. and Pottle, B., 2009, **Oracle Database 11g: SQL Fundamentals**, Vol.1, Edition 1.1, Oracle Corporation.

[3] Singh, P. and Pottle, B., 2009, **Oracle Database 11g: SQL Fundamentals**, Vol.2, Edition 1.1, Oracle Corporation.

ภาคผนวก

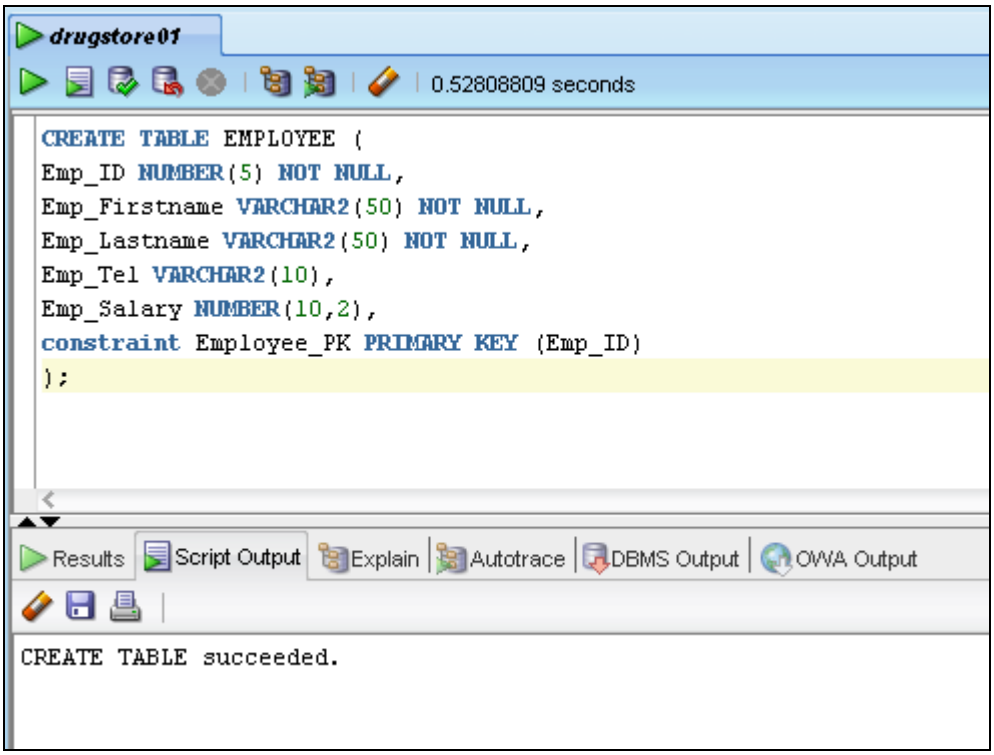
ก การสร้างตารางและเรียกดูโครงสร้างตารางในฐานข้อมูล

- **การสร้างตารางพนักงาน (EMPLOYEE)**

  - การใช้คำสั่ง CREATE เพื่อสร้างตารางพนักงาน (EMPLOYEE) ซึ่งมี SQL statement ดังนี้

```
CREATE TABLE EMPLOYEE (
Emp_ID NUMBER(5) NOT NULL,
Emp_Firstname VARCHAR2(50) NOT NULL,
Emp_Lastname VARCHAR2(50) NOT NULL,
Emp_Tel VARCHAR2(10),
Emp_Salary NUMBER(10,2),
constraint Employee_PK PRIMARY KEY (Emp_ID) );
```

  - สร้างตารางพนักงาน (EMPLOYEE) โดยใช้เครื่องมือ SQL Developer ดังรูปที่ ก.1



รูปที่ ก.1 สร้างตารางพนักงาน (EMPLOYEE)

จากรูปที่ ก.1 เป็นการสร้างตารางพนักงาน (EMPLOYEE) ซึ่งจะเก็บข้อมูลเกี่ยวกับพนักงานดังนี้

1. รหัสพนักงาน (Emp_ID) กำหนดให้ชนิดของข้อมูลเป็นตัวเลข (NUMBER) ขนาด 5 หลัก และเป็นคีย์หลัก (Primary key) ซึ่งห้ามมีค่าใดค่าหนึ่งเป็นค่าว่าง (Not null) หรือพนักงานทุกคนในตารางต้องมีรหัสพนักงาน

2. ชื่อพนักงาน (Emp_Firstname) กำหนดให้ชนิดของข้อมูลเป็นตัวอักษร (VARCHAR2) ขนาด 50 ตัวอักษร และห้ามมีค่าใดค่าหนึ่งเป็นค่าว่าง (Not null)

3. นามสกุลพนักงาน (Emp_Lastname) กำหนดให้ชนิดของข้อมูลเป็นตัวอักษร (VARCHAR2) ขนาด 50 ตัวอักษร และห้ามมีค่าใดค่าหนึ่งเป็นค่าว่าง (Not null)

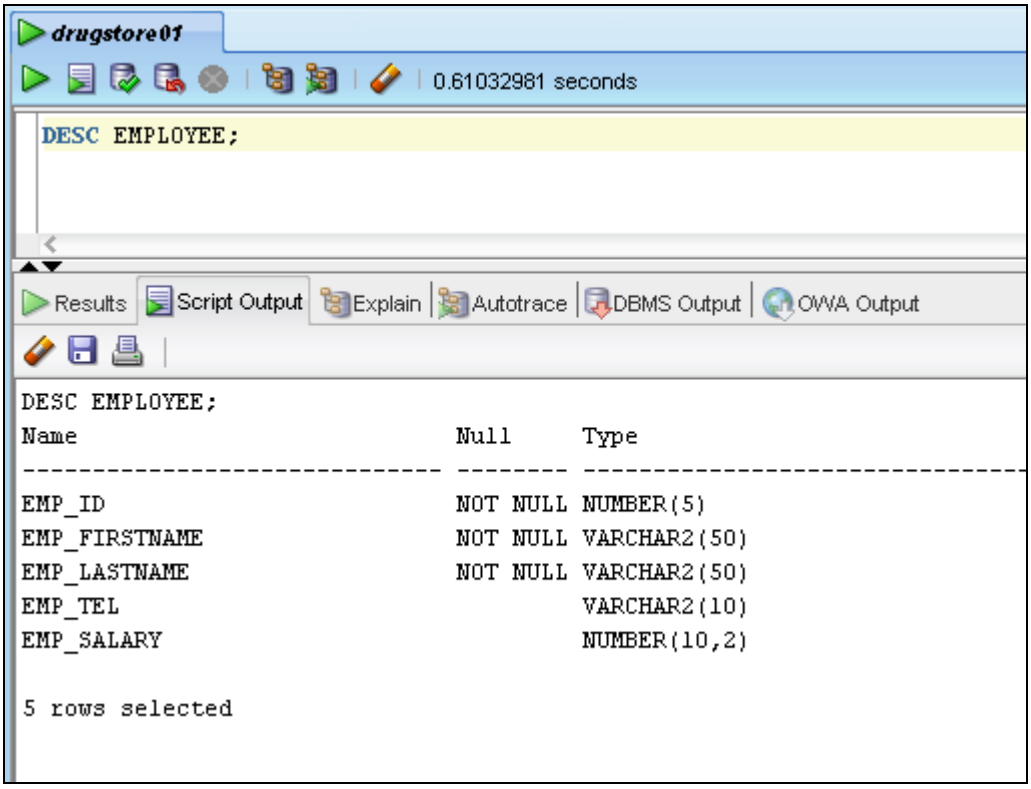
4. เบอร์โทรศัพท์ของพนักงาน (Emp_Tel) กำหนดให้ชนิดของข้อมูลเป็นตัวอักษร (VARCHAR2) ขนาด 10 ตัวอักษร

5. เงินเดือนของพนักงาน (Emp_Salary) กำหนดให้ชนิดของข้อมูลเป็นตัวเลข (NUMBER) ขนาด 10 หลัก และทศนิยม 2 ตำแหน่ง

- ใช้ใช้คำสั่ง DESC เพื่อเรียกดูโครงสร้างพนักงาน (EMPLOYEE) ซึ่งมี SQL statement ดังนี้

```
DESC EMPLOYEE;
```

- ใช้คำสั่ง DESC เพื่อเรียกดูโครงสร้างพนักงาน (EMPLOYEE) โดยใช้เครื่องมือ SQL Developer ดังรูปที่ ก.2



รูปที่ ก.2 เรียกดูโครงสร้างตารางพนักงาน (EMPLOYEE)

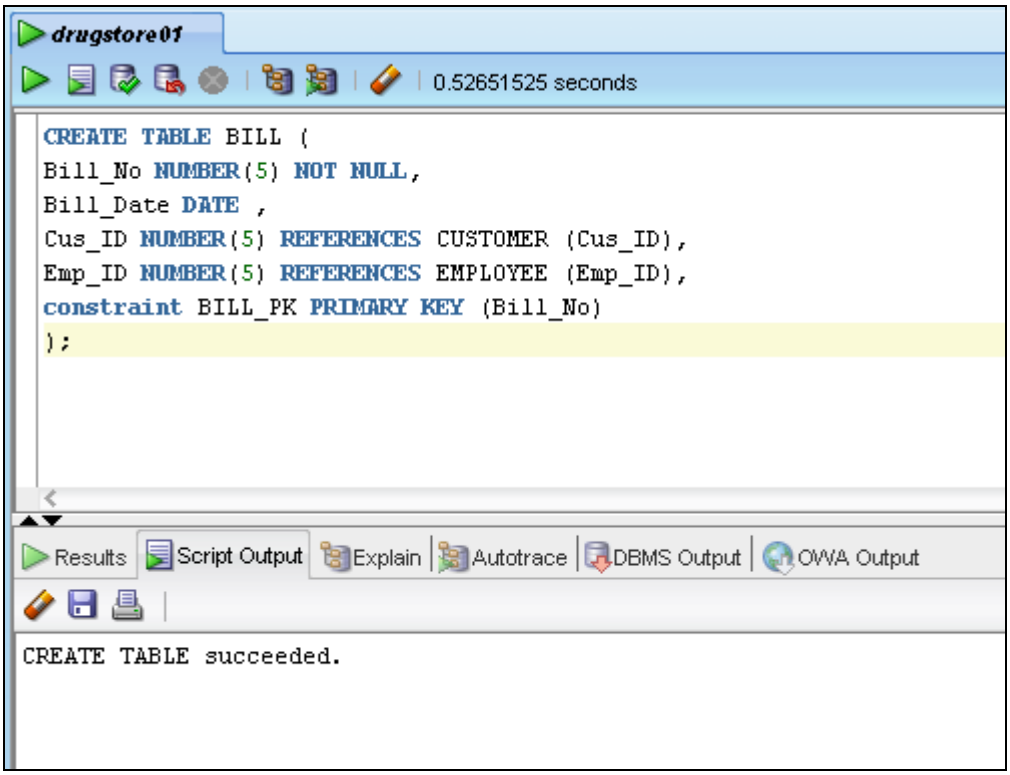
จากรูปที่ ก.2 เป็นการเรียกดูโครงสร้างตารางพนักงาน (EMPLOYEE) จะประกอบด้วยทั้งหมด 5 คอลัมน์ ได้แก่ รหัสพนักงาน ชื่อพนักงาน นามสกุลพนักงาน เบอร์โทรศัพท์ของพนักงาน และเงินเดือนของพนักงาน

- **การสร้างตารางใบเสร็จรับเงิน (BILL)**

  - การใช้คำสั่ง CREATE เพื่อสร้างตารางใบเสร็จรับเงิน (BILL) ซึ่งมี SQL statement ดังนี้

```
CREATE TABLE BILL (
Bill_No NUMBER(5) NOT NULL,
Bill_Date DATE ,
Cus_ID NUMBER(5) REFERENCES CUSTOMER (Cus_ID),
Emp_ID NUMBER(5) REFERENCES EMPLOYEE (Emp_ID),
constraint BILL_PK PRIMARY KEY (Bill_No) );
```

  - สร้างตารางใบเสร็จรับเงิน (BILL) โดยใช้เครื่องมือ SQL Developer ดังรูปที่ ก.3



รูปที่ ก.3 สร้างตารางใบเสร็จรับเงิน (BILL)

จากรูปที่ ก.3 เป็นการสร้างตารางใบเสร็จรับเงิน (BILL) ซึ่งจะเก็บข้อมูลเกี่ยวกับใบเสร็จรับเงิน ดังนี้

1. เลขที่ใบเสร็จรับเงิน (Bill_No) กำหนดให้ชนิดของข้อมูลเป็นตัวเลข (NUMBER) ขนาด 5 หลัก และเป็นคีย์หลัก (Primary key) ซึ่งห้ามมีค่าใดค่าหนึ่งเป็นค่าว่าง (Not null) หรือใบเสร็จทุกใบในตารางต้องมีเลขที่ใบเสร็จรับเงิน
2. วันที่ซื้อสินค้า (Bill_Date) กำหนดให้ชนิดของข้อมูลเป็นวันที่ (DATE)

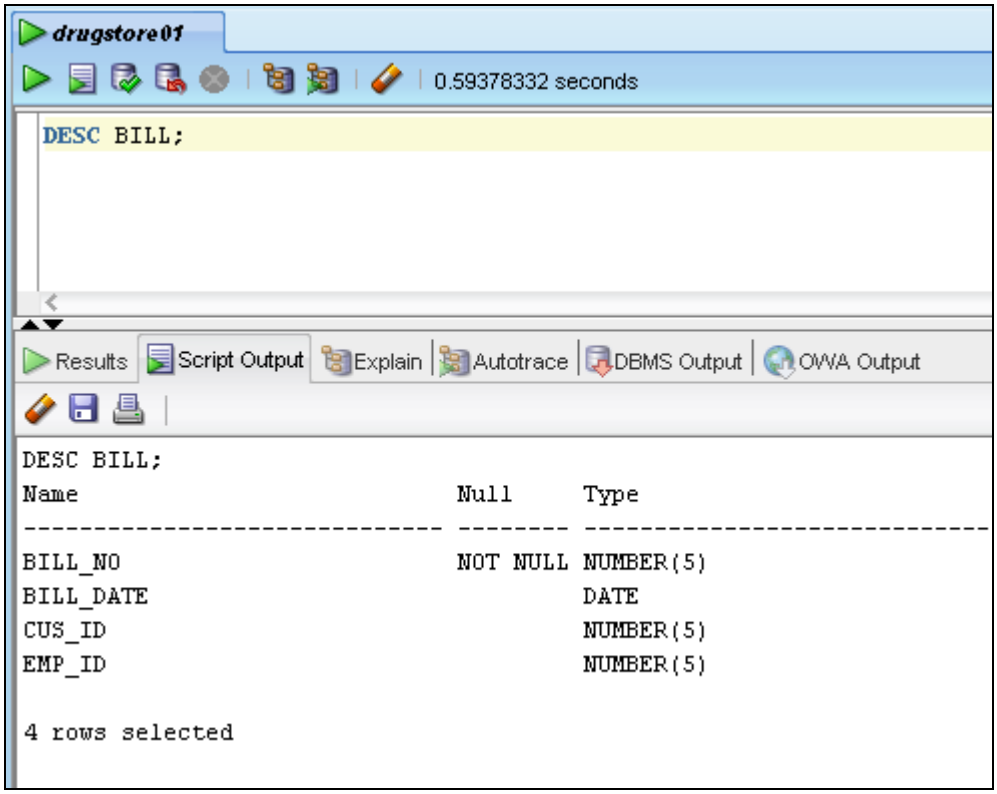
3. รหัสลูกค้า (Cus_ID) กำหนดให้ชนิดของข้อมูลเป็นตัวเลข (NUMBER) อ้างอิงมาจากตารางลูกค้า (CUSTOMER)

4. รหัสพนักงาน (Emp_ID) กำหนดให้ชนิดของข้อมูลเป็นตัวเลข (NUMBER) อ้างอิงมาจากตารางพนักงาน (EMPLOYEE)

- ใช้คำสั่ง DESC เพื่อเรียกดูโครงสร้างตารางใบเสร็จรับเงิน (Bill) ซึ่งมี SQL statement ดังนี้

```
DESC BILL;
```

- ใช้คำสั่ง DESC เพื่อเรียกดูโครงสร้างตารางใบเสร็จรับเงิน (Bill) โดยใช้เครื่องมือ SQL Developer ดังรูปที่ ก.4



รูปที่ ก.4 เรียกดูโครงสร้างตารางใบเสร็จรับเงิน (BILL)

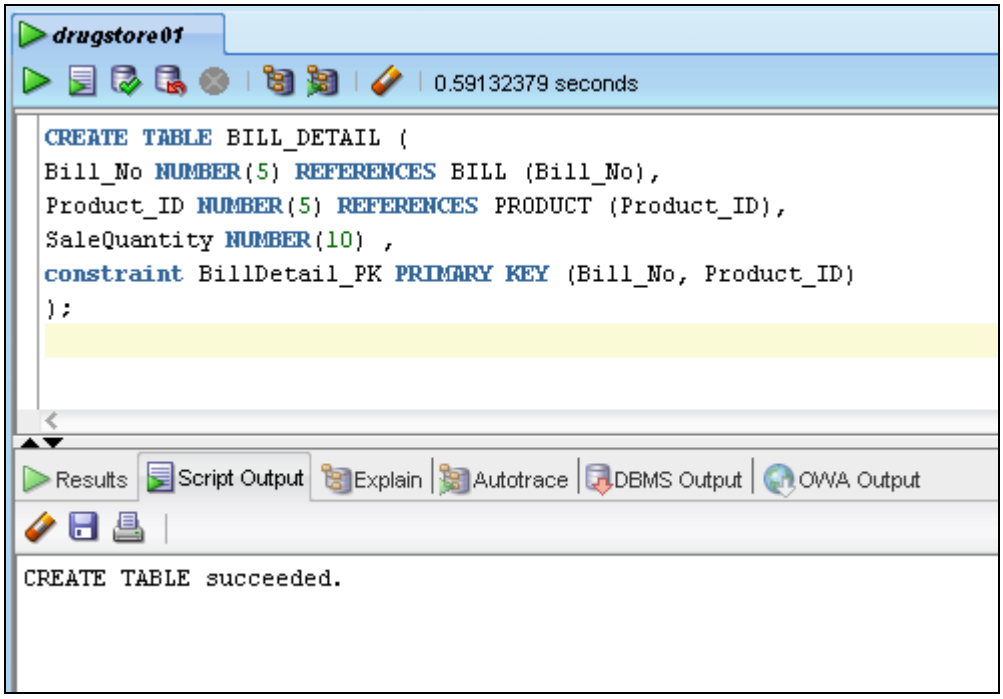
จากรูปที่ ก.4 เป็นการเรียกดูโครงสร้างตารางใบเสร็จรับเงิน (BILL) จะประกอบด้วยทั้งหมด 4 คอลัมน์ ได้แก่ เลขที่ใบเสร็จรับเงิน วันที่ซื้อสินค้า รหัสลูกค้า และรหัสพนักงาน

- **การสร้างตารางรายละเอียดใบเสร็จรับเงิน (BILL_DETAIL)**

- การใช้คำสั่ง CREATE เพื่อสร้างตารางรายละเอียดใบเสร็จรับเงิน (BILL_DETAIL) ซึ่งมี SQL statement ดังนี้

```
CREATE TABLE BILL_DETAIL (
Bill_No NUMBER(5) REFERENCES BILL (Bill_No),
Product_ID NUMBER(5) REFERENCES PRODUCT (Product_ID),
SaleQuantity NUMBER(10) ,
constraint BillDetail_PK PRIMARY KEY (Bill_No, Product_ID) );
```

- สร้างตารางรายละเอียดใบเสร็จรับเงิน (BILL_DETAIL) โดยใช้เครื่องมือ SQL Developer ดังรูปที่ ก.5



รูปที่ ก.5 สร้างตารางรายละเอียดใบเสร็จรับเงิน (BILL_DETAIL)

จากรูปที่ ก.5 เป็นการสร้างตารางรายละเอียดใบเสร็จรับเงิน (BILL_DETAIL) ซึ่งจะเก็บข้อมูลเกี่ยวกับรายละเอียดใบเสร็จรับเงินดังนี้

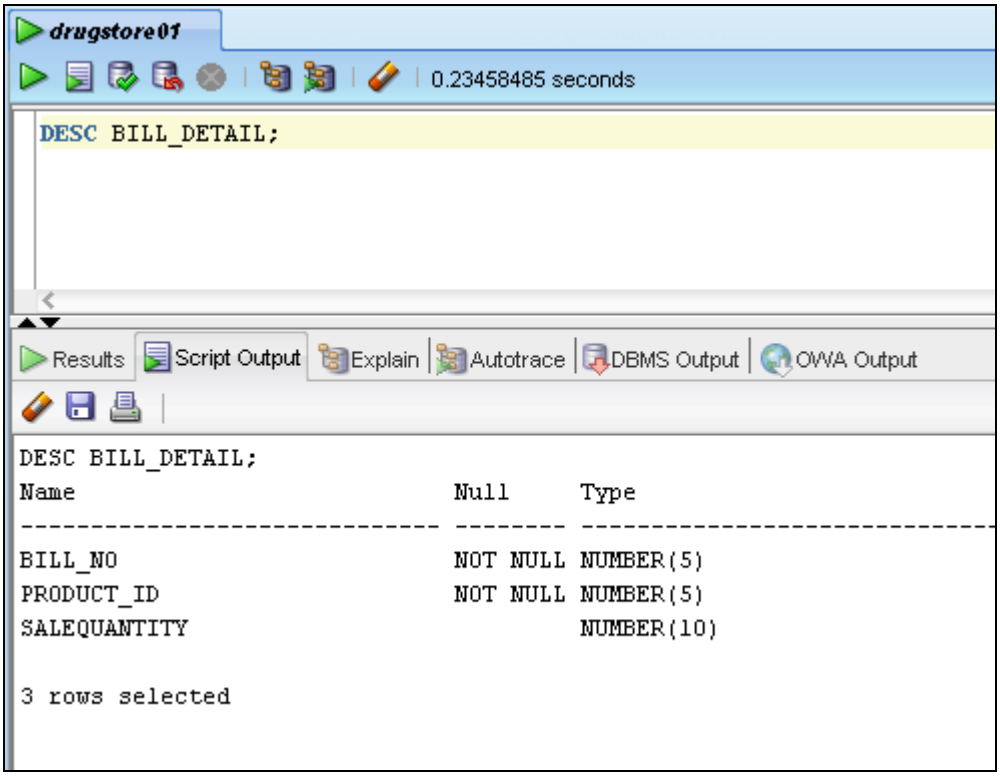
1. เลขที่ใบเสร็จรับเงิน (Bill_No) กำหนดให้ชนิดของข้อมูลเป็นตัวเลข (NUMBER) ขนาด 5 หลัก คอลัมน์นี้เป็นคีย์หลัก (Primary key) ซึ่งห้ามมีค่าใดค่าหนึ่งเป็นค่าว่าง (Not null) และเป็นคีย์นอก (Foreign key) อ้างอิงมาจากตารางใบเสร็จรับเงิน (BILL)
2. รหัสสินค้า (Product_ID) กำหนดให้ชนิดของข้อมูลเป็นตัวเลข (NUMBER) ขนาด 5 หลัก คอลัมน์เป็นคีย์หลัก (Primary key) ซึ่งห้ามมีค่าใดค่าหนึ่งเป็นค่าว่าง (Not null) และเป็นคีย์นอก (Foreign key) อ้างอิงมาจากตารางสินค้า (PRODUCT)

3. จำนวนสินค้าที่ขาย (Sale_Quantity) กำหนดให้ชนิดของข้อมูลเป็นตัวเลข (NUMBER) ขนาด 10 หลัก

- ใช้คำสั่ง DESC เพื่อเรียกดูโครงสร้างตารางรายละเอียดใบเสร็จรับเงิน (BILL_DETAIL) ซึ่งมี SQL statement ดังนี้

DESC BILL_DETAIL;

- ใช้คำสั่ง DESC เพื่อเรียกดูโครงสร้างตารางรายละเอียดใบเสร็จรับเงิน (BILL_DETAIL) โดยใช้เครื่องมือ SQL Developer ดังรูปที่ ก.6



รูปที่ ก.6 เรียกดูโครงสร้างตารางรายละเอียดใบเสร็จรับเงิน (BILL_DETAIL)

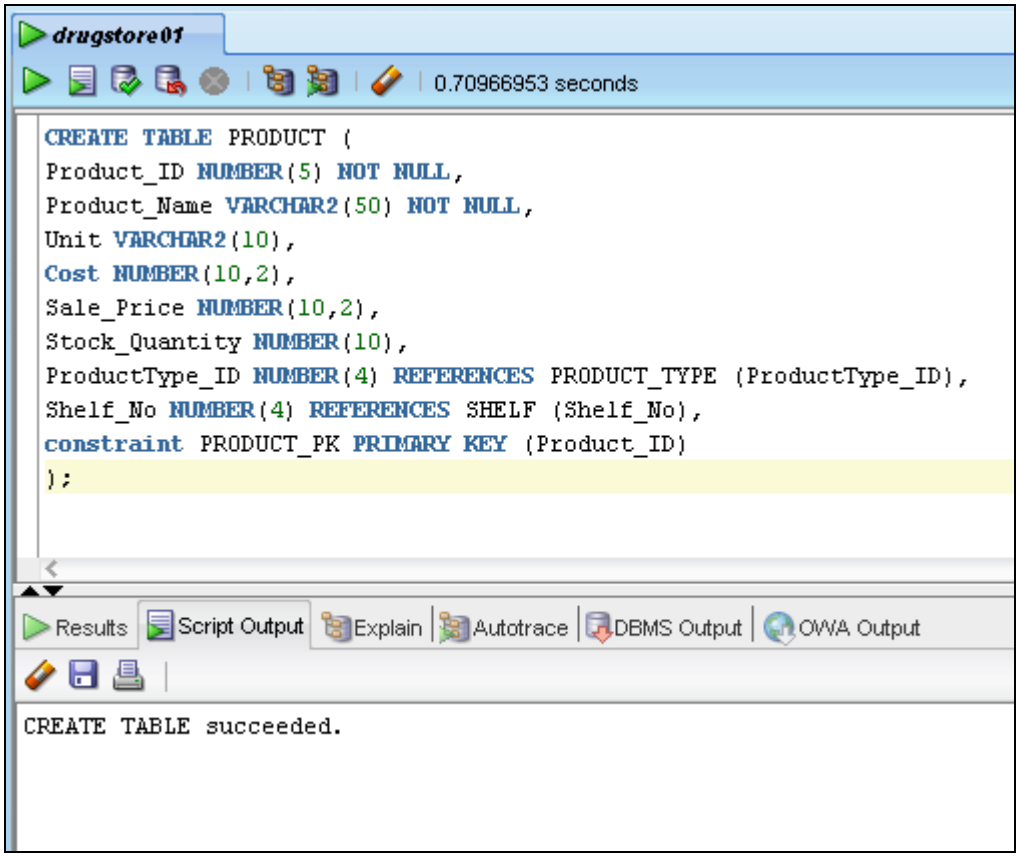
จากรูปที่ ก.6 เป็นการเรียกดูโครงสร้างตารางรายละเอียดใบเสร็จรับเงิน (BILL_DETAIL) จะประกอบด้วยทั้งหมด 3 คอลัมน์ ได้แก่ เลขที่ใบเสร็จรับเงิน รหัสสินค้า และจำนวนสินค้าที่ขาย

- **การสร้างตารางสินค้า (PRODUCT)**

  - การใช้คำสั่ง CREATE เพื่อสร้างตารางสินค้า (PRODUCT) ซึ่งมี SQL statement ดังนี้

```
CREATE TABLE PRODUCT (
Product_ID NUMBER(5) NOT NULL,
Product_Name VARCHAR2(50) NOT NULL,
Unit VARCHAR2(10),
Cost NUMBER(10,2),
Sale_Price NUMBER(10,2),
Stock_Quantity NUMBER(10),
ProductType_ID NUMBER(4) REFERENCES PRODUCT_TYPE
(ProductType_ID),
Shelf_No NUMBER(4) REFERENCES SHELF (Shelf_No),
constraint PRODUCT_PK PRIMARY KEY (Product_ID) );
```

  - สร้างตารางสินค้า (PRODUCT) โดยใช้เครื่องมือ SQL Developer ดังรูปที่ ก.7



รูปที่ ก.7 สร้างตารางสินค้า (PRODUCT)

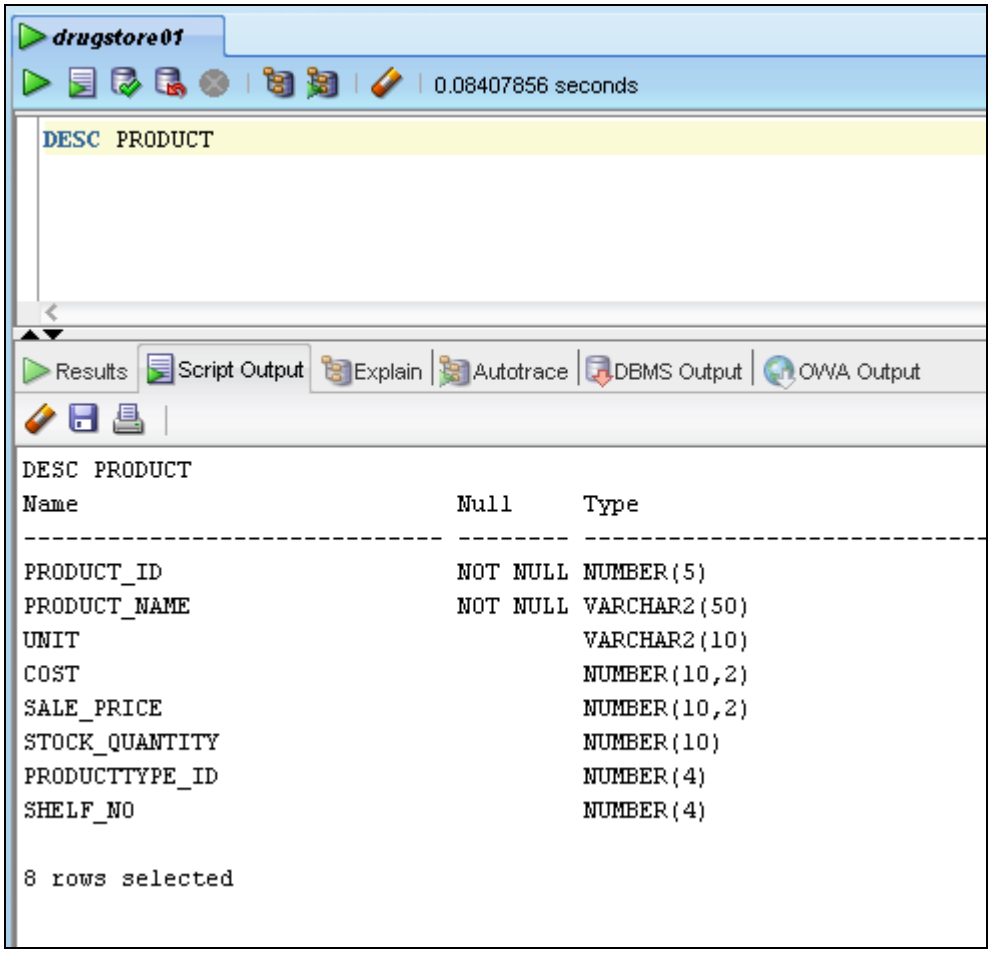
จากรูปที่ ก.7 เป็นการสร้างตารางสินค้า (PRODUCT) ซึ่งจะเก็บข้อมูลเกี่ยวกับสินค้าดังนี้

1. รหัสสินค้า (Product_ID) กำหนดให้ชนิดของข้อมูลเป็นตัวเลข (NUMBER) ขนาด 5 หลัก และเป็นคีย์หลัก (Primary key) ซึ่งห้ามมีค่าใดค่าหนึ่งเป็นค่าว่าง (Not null) หรือสินค้าทุกชิ้นในตารางต้องมีรหัสสินค้า
2. ชื่อสินค้า (Product_Name) กำหนดให้ชนิดของข้อมูลเป็นตัวอักษร (VARCHAR2) ขนาด 50 ตัวอักษร และห้ามมีค่าใดค่าหนึ่งเป็นค่าว่าง (Not null)
3. หน่วยของสินค้า (Unit) กำหนดให้ชนิดของข้อมูลเป็นตัวอักษร (VARCHAR2) ขนาด 10 ตัวอักษร
4. ต้นทุนต่อหน่วย (Cost) กำหนดให้ชนิดของข้อมูลเป็นตัวเลข (NUMBER) ขนาด 10 หลักและทศนิยม 2 ตำแหน่ง
5. ราคาขายต่อหน่วย (Sale_Price) กำหนดให้ชนิดของข้อมูลเป็นตัวเลข (NUMBER) ขนาด 10 หลัก และทศนิยม 2 ตำแหน่ง
6. จำนวนสินค้าในคลังสินค้า (Stock_Quantity) กำหนดให้ชนิดของข้อมูลเป็นตัวเลข (NUMBER) ขนาด 10 หลัก
7. รหัสประเภทสินค้า (ProductType_ID) กำหนดให้ชนิดของข้อมูลเป็นตัวเลข (NUMBER) อ้างอิงมาจากตารางประเภทสินค้า (PRODUCT_TYPE)
8. หมายเลขชั้นวางสินค้า (Shelf_No) กำหนดให้ชนิดของข้อมูลเป็นตัวเลข (NUMBER) อ้างอิงมาจากตารางชั้นวางสินค้า (SHELF)

- ใช้คำสั่ง DESC เพื่อเรียกดูโครงสร้างตารางสินค้า (PRODUCT) ซึ่งมี SQL statement ดังนี้

```
DESC PRODUCT;
```

- ใช้คำสั่ง DESC เพื่อเรียกดูโครงสร้างตารางสินค้า (PRODUCT) โดยใช้เครื่องมือ SQL Developer ดังรูปที่ ก.8



รูปที่ ก.8 เรียกดูโครงสร้างตารางสินค้า (PRODUCT)

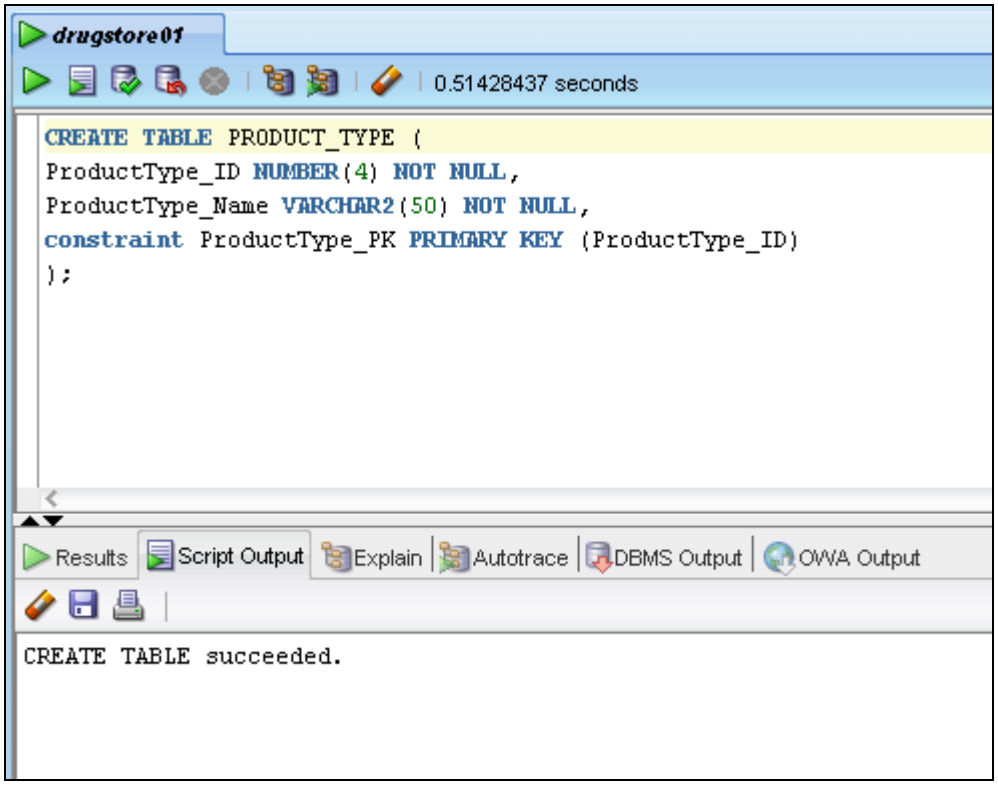
จากรูปที่ ก.8 เป็นการเรียกดูโครงสร้างตารางสินค้า (PRODUCT) จะประกอบด้วยทั้งหมด 8 คอลัมน์ ได้แก่ รหัสสินค้า ชื่อสินค้า หน่วยของสินค้า ต้นทุนต่อหน่วย ราคาขายต่อหน่วย จำนวนสินค้าในคลังสินค้า รหัสประเภทสินค้า และหมายเลขชั้นวางสินค้า

- **การสร้างตารางประเภทสินค้า (PRODUCT_TYPE)**

- การใช้คำสั่ง CREATE เพื่อสร้างตารางประเภทสินค้า (PRODUCT_TYPE) ซึ่งมี SQL statement ดังนี้

```
CREATE TABLE PRODUCT_TYPE (
ProductType_ID NUMBER(4) NOT NULL,
ProductType_Name VARCHAR2(50) NOT NULL,
constraint ProductType_PK PRIMARY KEY (ProductType_ID) );
```

- สร้างตารางลูกค้า (PRODUCT_TYPE) โดยใช้เครื่องมือ SQL Developer ดังรูปที่ ก.9



รูปที่ ก.9 สร้างตารางประเภทสินค้า (PRODUCT_TYPE)

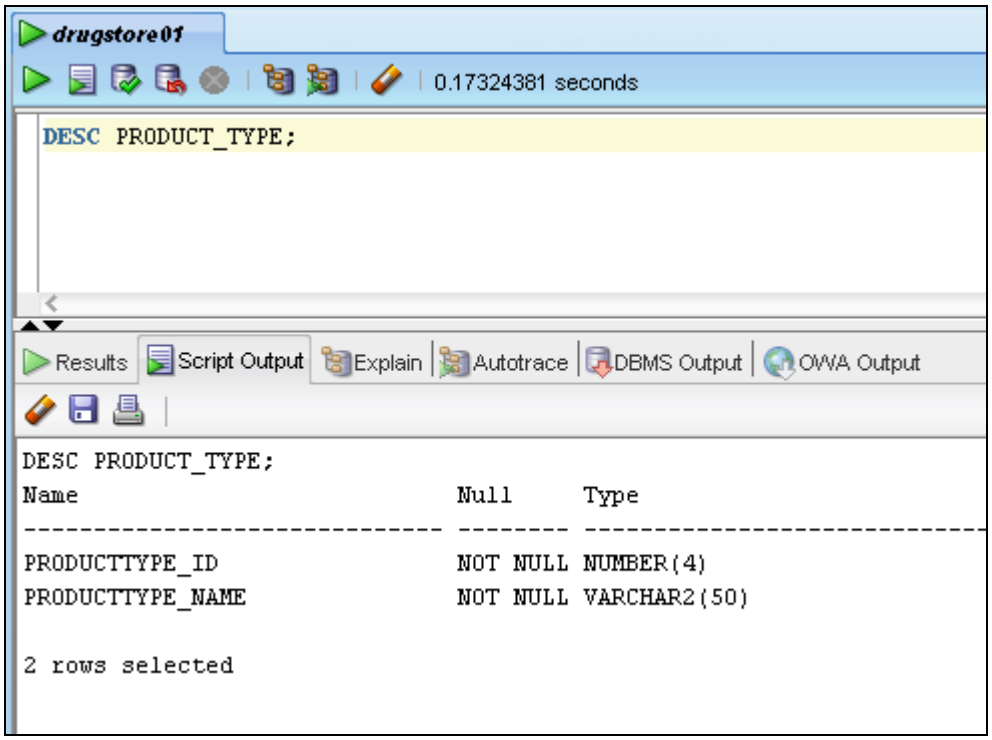
จากรูปที่ ก.9 เป็นการสร้างตารางประเภทสินค้า (PRODUCT_TYPE) ซึ่งจะเก็บข้อมูลเกี่ยวกับประเภทสินค้าดังนี้

1. รหัสประเภทสินค้า (ProductType_ID) กำหนดให้ชนิดของข้อมูลเป็นตัวเลข (NUMBER) ขนาด 4 หลัก และเป็นคีย์หลัก (Primary key) ซึ่งห้ามมีค่าใดค่าหนึ่งเป็นค่าว่าง (Not null) หรือประเภทสินค้าทุกคนชนิดในตารางต้องมีรหัสประเภทสินค้า
2. ชื่อประเภทสินค้า (ProductType_Name) กำหนดให้ชนิดของข้อมูลเป็นตัวอักษร (VARCHAR2) ขนาด 50 ตัวอักษร และห้ามมีค่าใดค่าหนึ่งเป็นค่าว่าง (Not null)

- ใช้คำสั่ง DESC เพื่อเรียกดูโครงสร้างตารางประเภทสินค้า (PRODUCT_TYPE) ซึ่งมี SQL statement ดังนี้

```
DESC PRODUCT_TYPE;
```

- ใช้คำสั่ง DESC เพื่อเรียกดูโครงสร้างตารางประเภทสินค้า (PRODUCT_TYPE) โดยใช้เครื่องมือ SQL Developer ดังรูปที่ ก.10



รูปที่ ก.10 เรียกดูโครงสร้างตารางประเภทสินค้า (PRODUCT_TYPE)

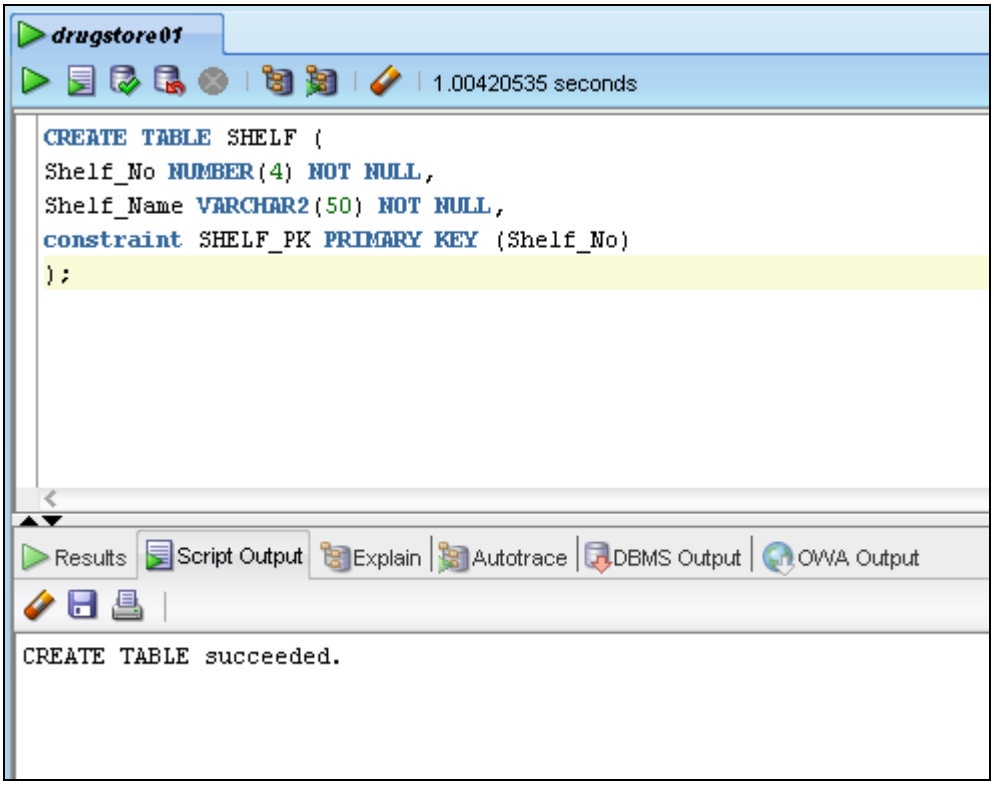
จากรูปที่ ก.10 เป็นการเรียกดูโครงสร้างตารางประเภทสินค้า (PRODUCT_TYPE) จะประกอบด้วยทั้งหมด 2 คอลัมน์ ได้แก่ รหัสประเภทสินค้า และชื่อประเภทสินค้า

- **การสร้างตารางชั้นวางสินค้า (SHELF)**

  - การใช้คำสั่ง CREATE เพื่อสร้างตารางชั้นวางสินค้า (SHELF) ซึ่งมี SQL statement ดังนี้

```
CREATE TABLE SHELF (
Shelf_No NUMBER(4) NOT NULL,
Shelf_Name VARCHAR2(50) NOT NULL,
constraint SHELF_PK PRIMARY KEY (Shelf_No) );
```

  - สร้างตารางชั้นวางสินค้า (SHELF) โดยใช้เครื่องมือ SQL Developer ดังรูปที่ ก.11



รูปที่ ก.11 สร้างตารางชั้นวางสินค้า (SHELF)

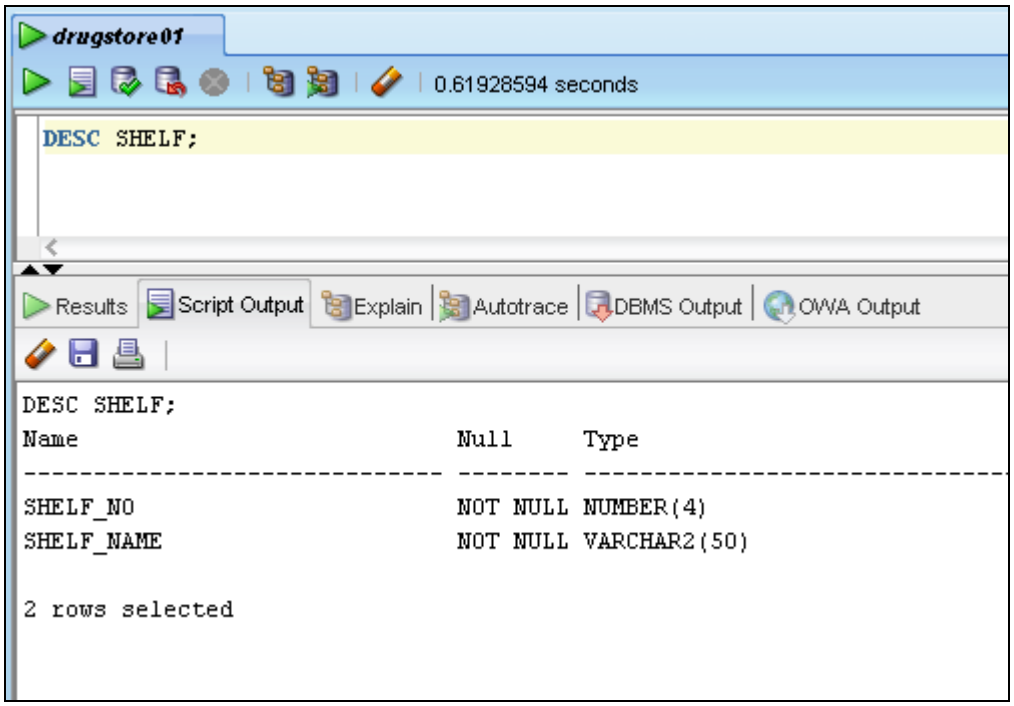
จากรูปที่ ก.11 เป็นการสร้างตารางชั้นวางสินค้า (SHELF) ซึ่งจะเก็บข้อมูลเกี่ยวกับชั้นวางสินค้า ดังนี้

1. หมายเลขชั้นวางสินค้า (Shelf_No) กำหนดให้ชนิดของข้อมูลเป็นตัวเลข (NUMBER) ขนาด 4 หลัก และเป็นคีย์หลัก (Primary key) ซึ่งห้ามมีค่าใดค่าหนึ่งเป็นค่าว่าง (Not null) หรือชั้นวางสินค้าทุกชั้นในตารางต้องมีหมายเลขชั้นวางสินค้า
2. ชื่อชั้นวางสินค้า (Shelf_Name) กำหนดให้ชนิดของข้อมูลเป็นตัวอักษร (VARCHAR2) ขนาด 50 ตัวอักษร และห้ามมีค่าใดค่าหนึ่งเป็นค่าว่าง (Not null)

- ใช้คำสั่ง DESC เพื่อเรียกดูโครงสร้างตารางชั้นวางสินค้า (SHELF) ซึ่งมี SQL statement ดังนี้

```
DESC SHELF;
```

- ใช้คำสั่ง DESC เพื่อเรียกดูโครงสร้างตารางชั้นวางสินค้า (SHELF) โดยใช้เครื่องมือ SQL Developer ดังรูปที่ ก.10



รูปที่ ก.12 เรียกดูโครงสร้างตารางชั้นวางสินค้า (SHELF)

จากรูปที่ ก.12 เป็นการเรียกดูโครงสร้างตารางชั้นวางสินค้า (SHELF) จะประกอบด้วยทั้งหมด 2 คอลัมน์ ได้แก่ หมายเลขชั้นวางสินค้า และชื่อชั้นวางสินค้า

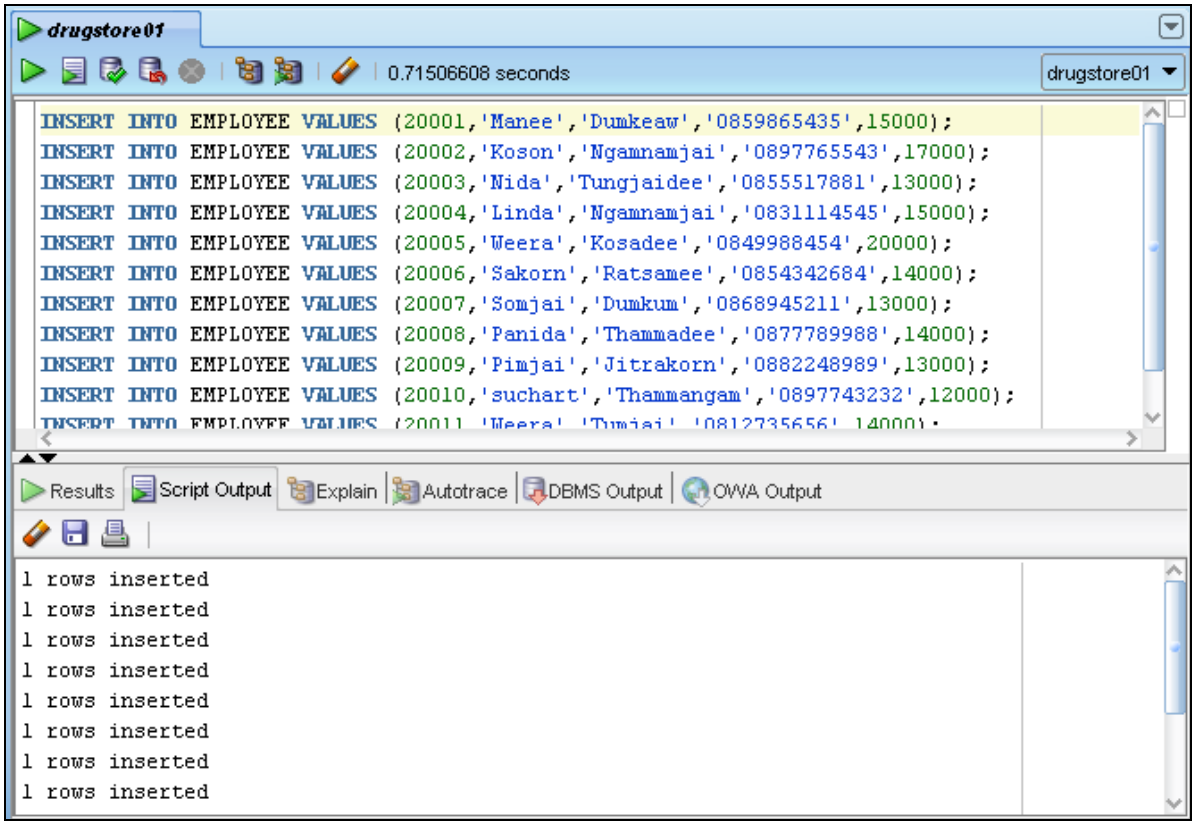
ข การเพิ่มข้อมูลในตาราง

- **การเพิ่มข้อมูลในตารางพนักงาน (EMPLOYEE)**

- การใช้คำสั่ง INSERT INTO เพื่อเพิ่มข้อมูลในตารางพนักงาน (EMPLOYEE) ซึ่งมี SQL statement ดังนี้

```
INSERT INTO EMPLOYEE VALUES
(20001,'Manee','Dumkeaw','0859865435',15000);
INSERT INTO EMPLOYEE VALUES
(20002,'Koson','Ngamnamjai','0897765543',17000);
INSERT INTO EMPLOYEE VALUES
(20003,'Nida','Tungjaidee','0855517881',13000);
```

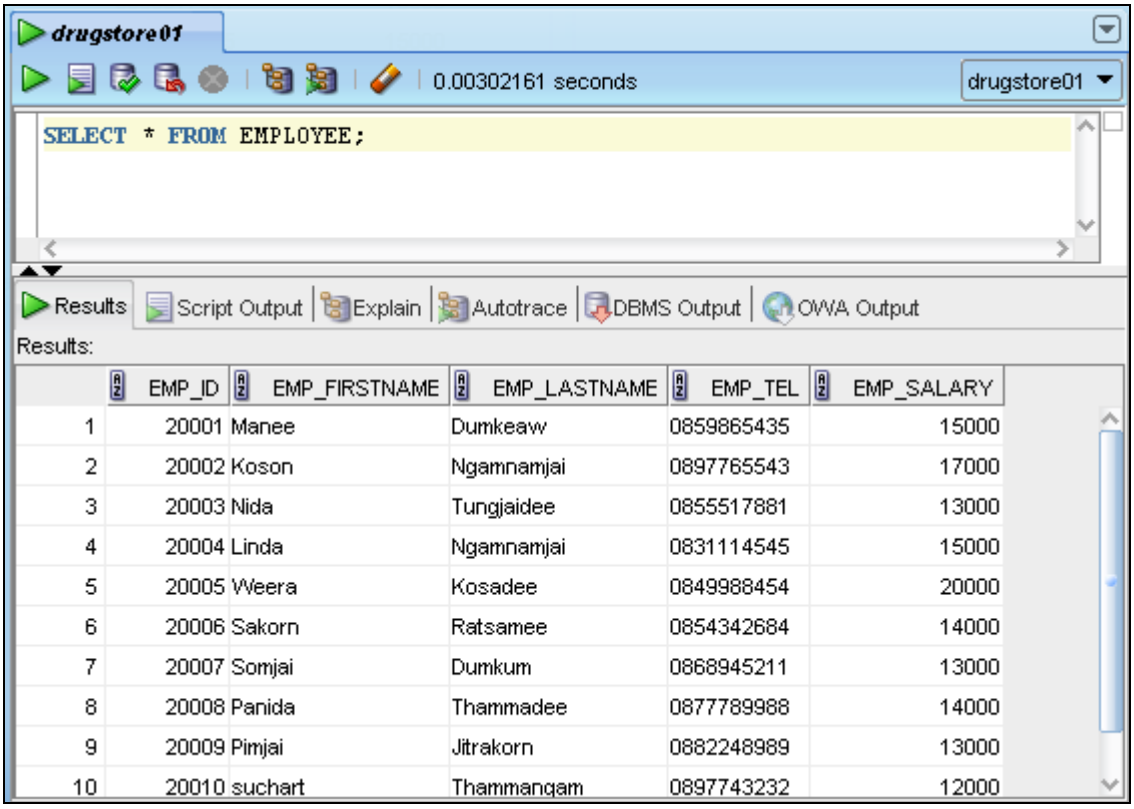
- ใช้คำสั่ง INSERT INTO เพื่อเพิ่มข้อมูลในตารางพนักงาน (EMPLOYEE) โดยใช้เครื่องมือ SQL Developer ดังรูปที่ ข.1



รูปที่ ข.1 เพิ่มข้อมูลในตารางพนักงาน (EMPLOYEE)

จากรูปที่ ข.1 เป็นการเพิ่มข้อมูลเข้าสู่ตารางพนักงาน (EMPLOYEE) ซึ่งต้องใส่ข้อมูลดังนี้ รหัสพนักงาน (Emp_ID) ชื่อพนักงาน (Emp_Firstname) นามสกุลพนักงาน (Emp_Lastname) เบอร์โทรศัพท์ของพนักงาน (Emp_Tel) และเงินเดือนของพนักงาน (Emp_Salary)

หลังจากที่ใส่ข้อมูลลงในตารางเรียบร้อยแล้ว สามารถดูข้อมูลในตารางพนักงาน (EMPLOYEE) ดังรูปที่ ข.2



	EMP_ID	EMP_FIRSTNAME	EMP_LASTNAME	EMP_TEL	EMP_SALARY
1	20001	Manee	Dumkeaw	0859865435	15000
2	20002	Koson	Ngamnamjai	0897765543	17000
3	20003	Nida	Tungjaidee	0855517881	13000
4	20004	Linda	Ngamnamjai	0831114545	15000
5	20005	Weera	Kosadee	0849988454	20000
6	20006	Sakorn	Ratsamee	0854342684	14000
7	20007	Somjai	Dumkum	0868945211	13000
8	20008	Panida	Thammadee	0877789988	14000
9	20009	Pimjai	Jitrakorn	0882248989	13000
10	20010	suchart	Thammangam	0897743232	12000

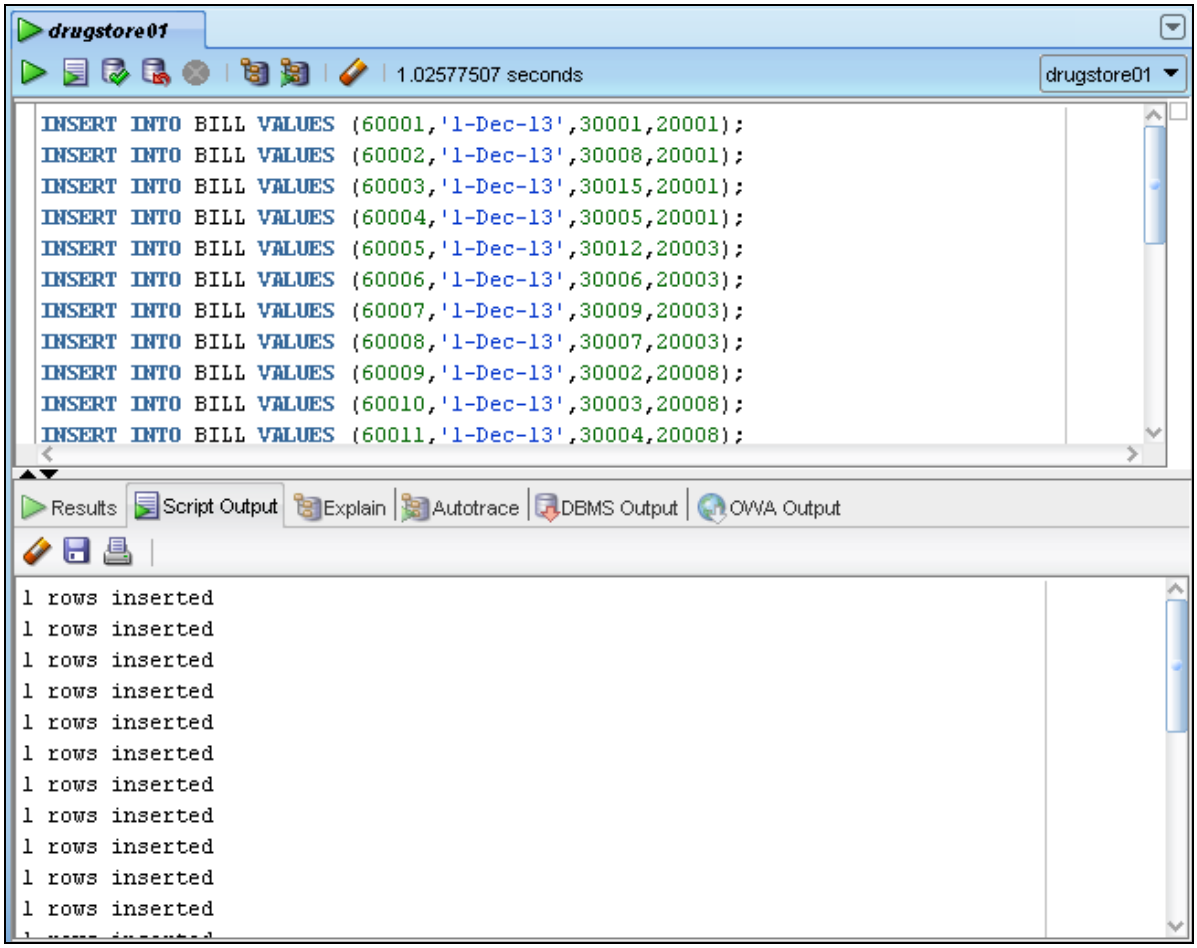
รูปที่ ข.2 ข้อมูลที่เพิ่มในตารางพนักงาน (EMPLOYEE)

- **การเพิ่มข้อมูลในตารางใบเสร็จรับเงิน (BILL)**

- การใช้คำสั่ง INSERT INTO เพื่อเพิ่มข้อมูลในตารางใบเสร็จรับเงิน (BILL) ซึ่งมี SQL statement ดังนี้

```
INSERT INTO BILL VALUES (60001,'1-Dec-13',30001,20001);
INSERT INTO BILL VALUES (60002,'1-Dec-13',30008,20001);
INSERT INTO BILL VALUES (60003,'1-Dec-13',30015,20001);
INSERT INTO BILL VALUES (60004,'1-Dec-13',30005,20001);
INSERT INTO BILL VALUES (60005,'1-Dec-13',30012,20003);
```

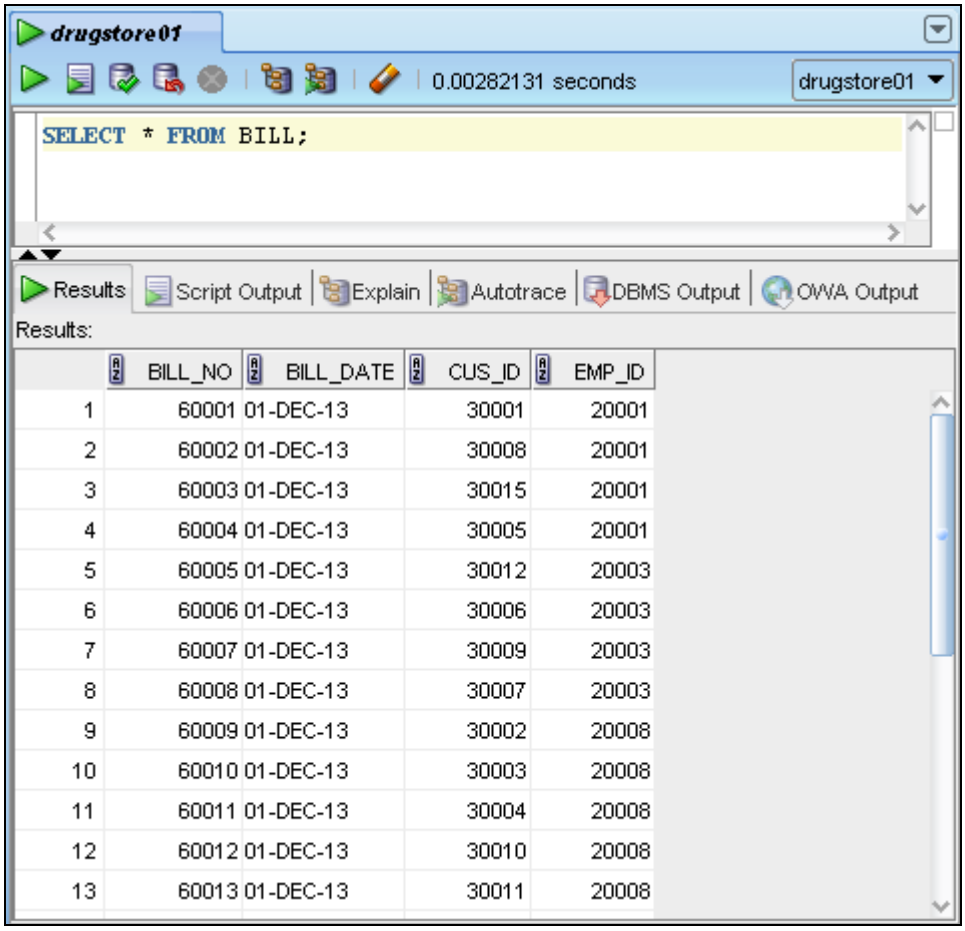
- ใช้คำสั่ง INSERT INTO เพื่อเพิ่มข้อมูลในใบเสร็จรับเงิน (BILL) โดยใช้เครื่องมือ SQL Developer ดังรูปที่ ข.3



รูปที่ ข.3 เพิ่มข้อมูลในตารางใบเสร็จรับเงิน (BILL)

จากรูปที่ ข.3 เป็นการเพิ่มข้อมูลเข้าสู่ตารางใบเสร็จรับเงิน (BILL) ซึ่งต้องใส่ข้อมูลดังนี้ เลขที่ใบเสร็จรับเงิน (Bill_No) วันที่ซื้อสินค้า (Bill_Date) รหัสลูกค้า (Cus_ID) และรหัสพนักงาน (Emp_ID)

หลังจากที่ใส่ข้อมูลลงในตารางเรียบร้อยแล้ว สามารถดูข้อมูลในตารางใบเสร็จรับเงิน (BILL) ดังรูปที่ ข.4



	BILL_NO	BILL_DATE	CUS_ID	EMP_ID
1	60001	01-DEC-13	30001	20001
2	60002	01-DEC-13	30008	20001
3	60003	01-DEC-13	30015	20001
4	60004	01-DEC-13	30005	20001
5	60005	01-DEC-13	30012	20003
6	60006	01-DEC-13	30006	20003
7	60007	01-DEC-13	30009	20003
8	60008	01-DEC-13	30007	20003
9	60009	01-DEC-13	30002	20008
10	60010	01-DEC-13	30003	20008
11	60011	01-DEC-13	30004	20008
12	60012	01-DEC-13	30010	20008
13	60013	01-DEC-13	30011	20008

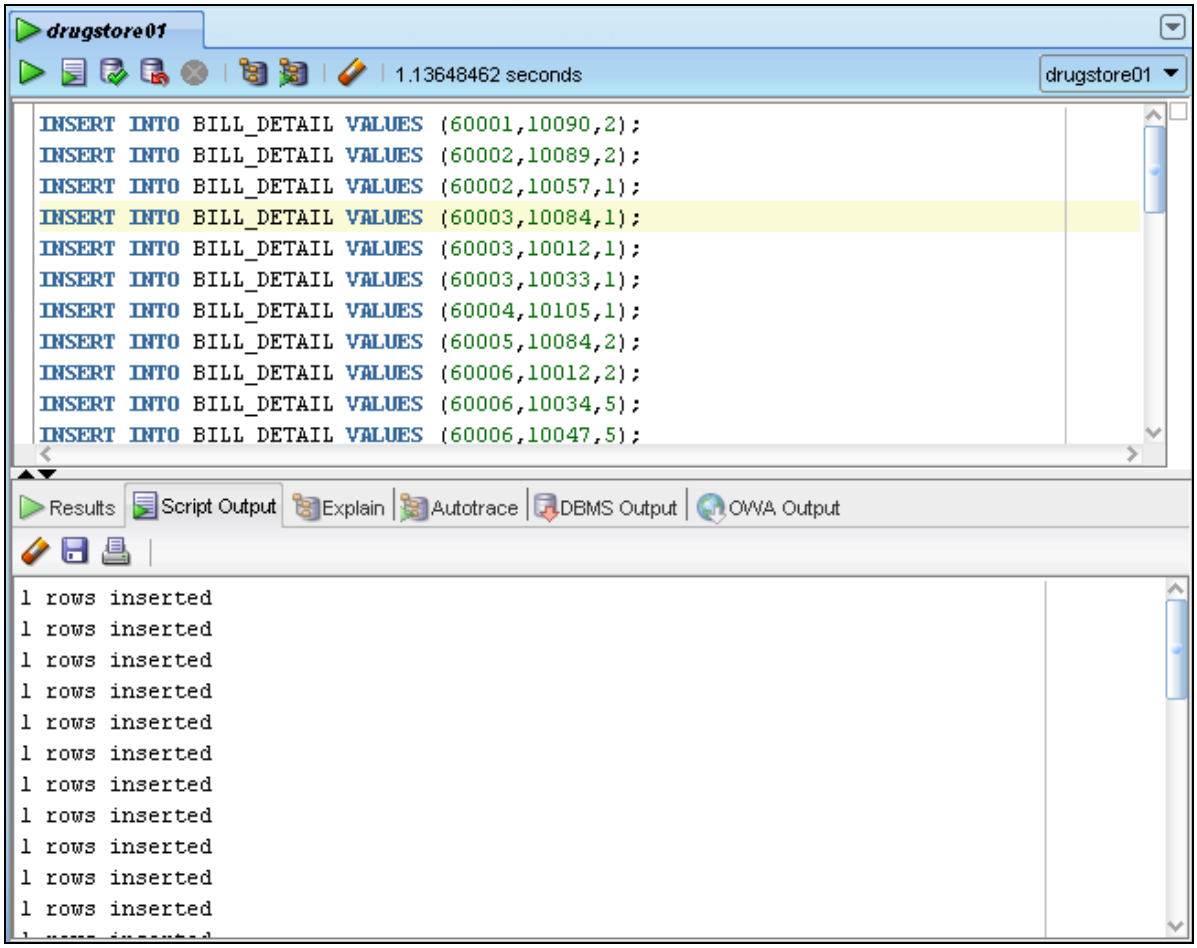
รูปที่ ข.4 ข้อมูลที่เพิ่มในตารางใบเสร็จรับเงิน (BILL)

- **การเพิ่มข้อมูลในตารางรายละเอียดใบเสร็จรับเงิน (BILL_DETAIL)**

- การใช้คำสั่ง INSERT INTO เพื่อเพิ่มข้อมูลในตารางรายละเอียดใบเสร็จรับเงิน (BILL_DETAIL) ซึ่งมี SQL statement ดังนี้

```
INSERT INTO BILL_DETAIL VALUES (60001,10090,2);
INSERT INTO BILL_DETAIL VALUES (60002,10089,2);
INSERT INTO BILL_DETAIL VALUES (60002,10057,1);
```

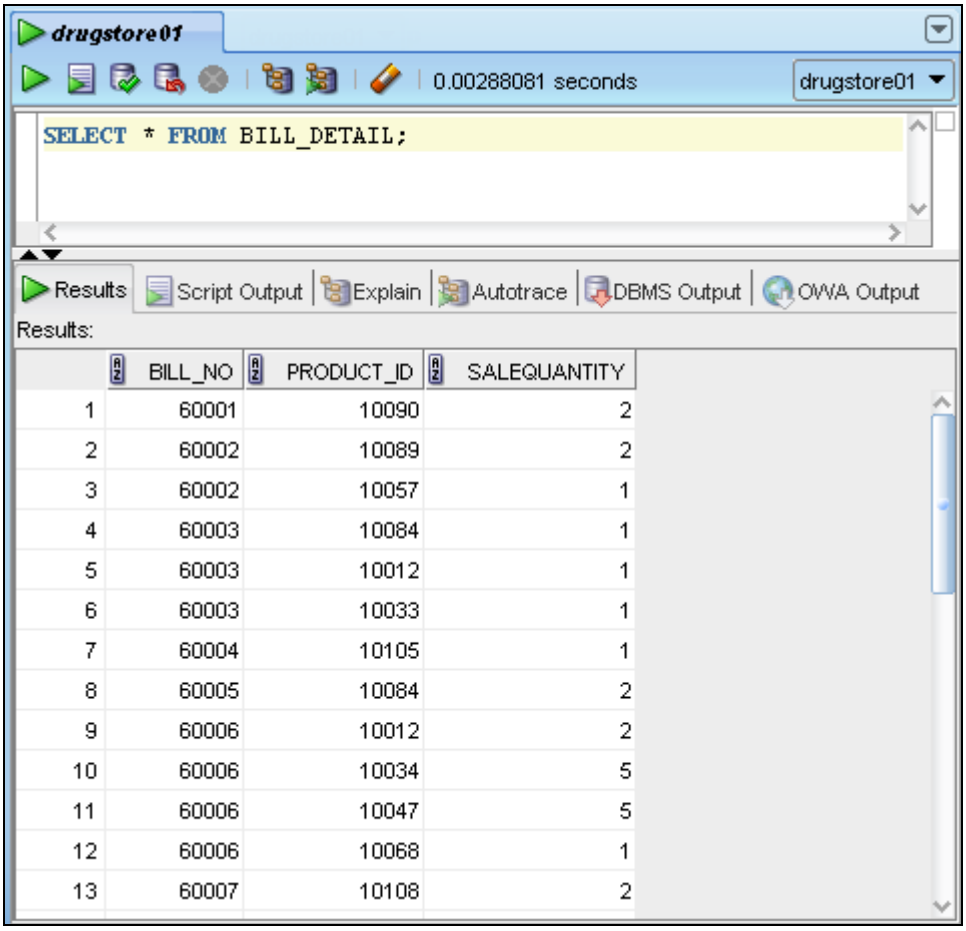
- ใช้คำสั่ง INSERT INTO เพื่อเพิ่มข้อมูลในตารางรายละเอียดใบเสร็จรับเงิน (BILL_DETAIL) โดยใช้เครื่องมือ SQL Developer ดังรูปที่ ข.5



รูปที่ ข.5 เพิ่มข้อมูลในตารางรายละเอียดใบเสร็จรับเงิน (BILL_DETAIL)

จากรูปที่ ข.5 เป็นการเพิ่มข้อมูลเข้าสู่ตารางรายละเอียดใบเสร็จรับเงิน (BILL_DETAIL) ซึ่งต้องใส่ข้อมูลดังนี้ เลขที่ใบเสร็จรับเงิน (Bill_No) รหัสสินค้า (Product_ID) และจำนวนสินค้าที่ขาย (Sale_Quantity)

หลังจากที่ใส่ข้อมูลลงในตารางเรียบร้อยแล้ว สามารถดูข้อมูลในตารางรายละเอียดใบเสร็จรับเงิน (BILL_DETAIL) ดังรูปที่ ข.6



	BILL_NO	PRODUCT_ID	SALEQUANTITY
1	60001	10090	2
2	60002	10089	2
3	60002	10057	1
4	60003	10084	1
5	60003	10012	1
6	60003	10033	1
7	60004	10105	1
8	60005	10084	2
9	60006	10012	2
10	60006	10034	5
11	60006	10047	5
12	60006	10068	1
13	60007	10108	2

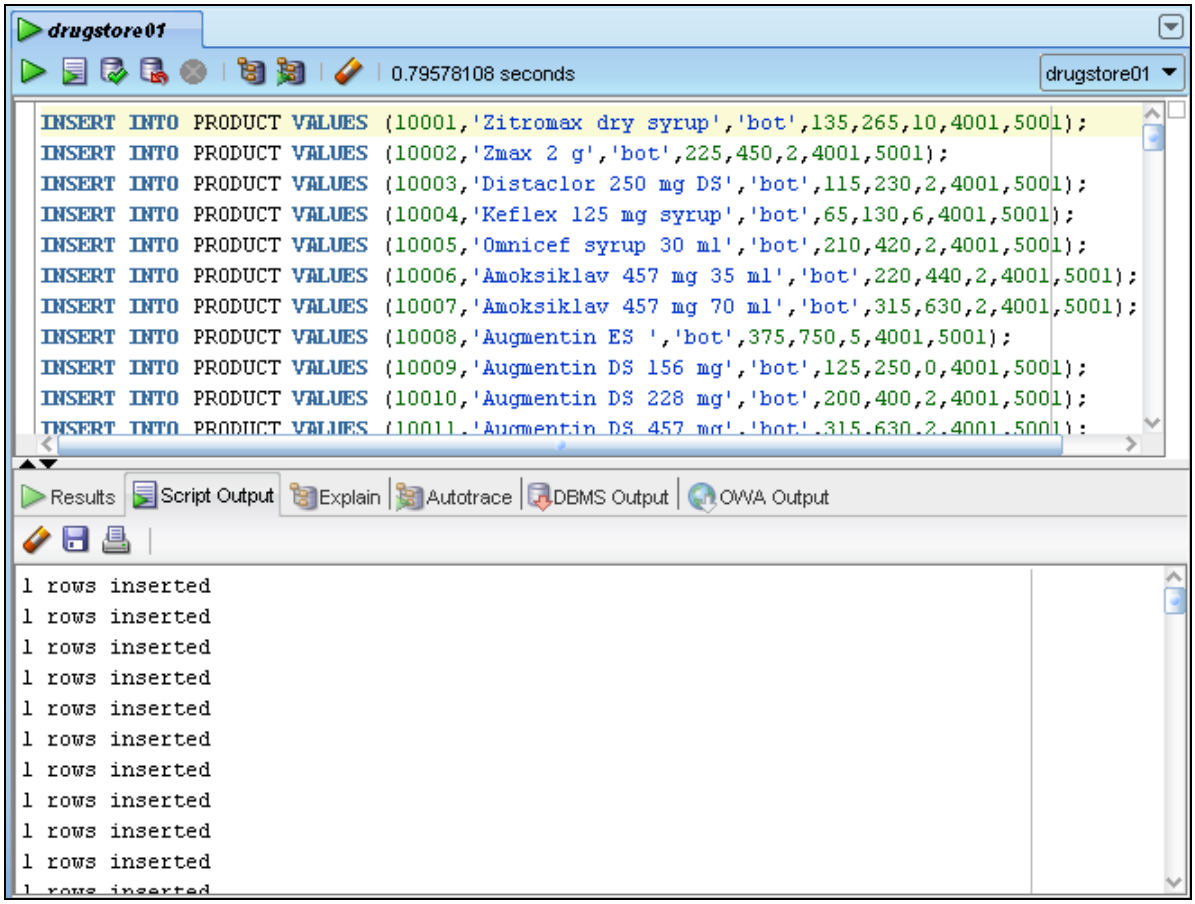
รูปที่ ข.6 ข้อมูลที่เพิ่มในตารางรายละเอียดใบเสร็จรับเงิน (BILL_DETAIL)

- **การเพิ่มข้อมูลในตารางสินค้า (PRODUCT)**

- การใช้คำสั่ง INSERT INTO เพื่อเพิ่มข้อมูลในตารางสินค้า (PRODUCT) ซึ่งมี SQL statement ดังนี้

```
INSERT INTO PRODUCT VALUES (10001,'Zitromax dry
syrup','bot',135,265,10,4001,5001);
INSERT INTO PRODUCT VALUES (10002,'Zmax 2
g','bot',225,450,2,4001,5001);
INSERT INTO PRODUCT VALUES (10003,'Distaclor 250 mg
DS','bot',115,230,2,4001,5001);
```

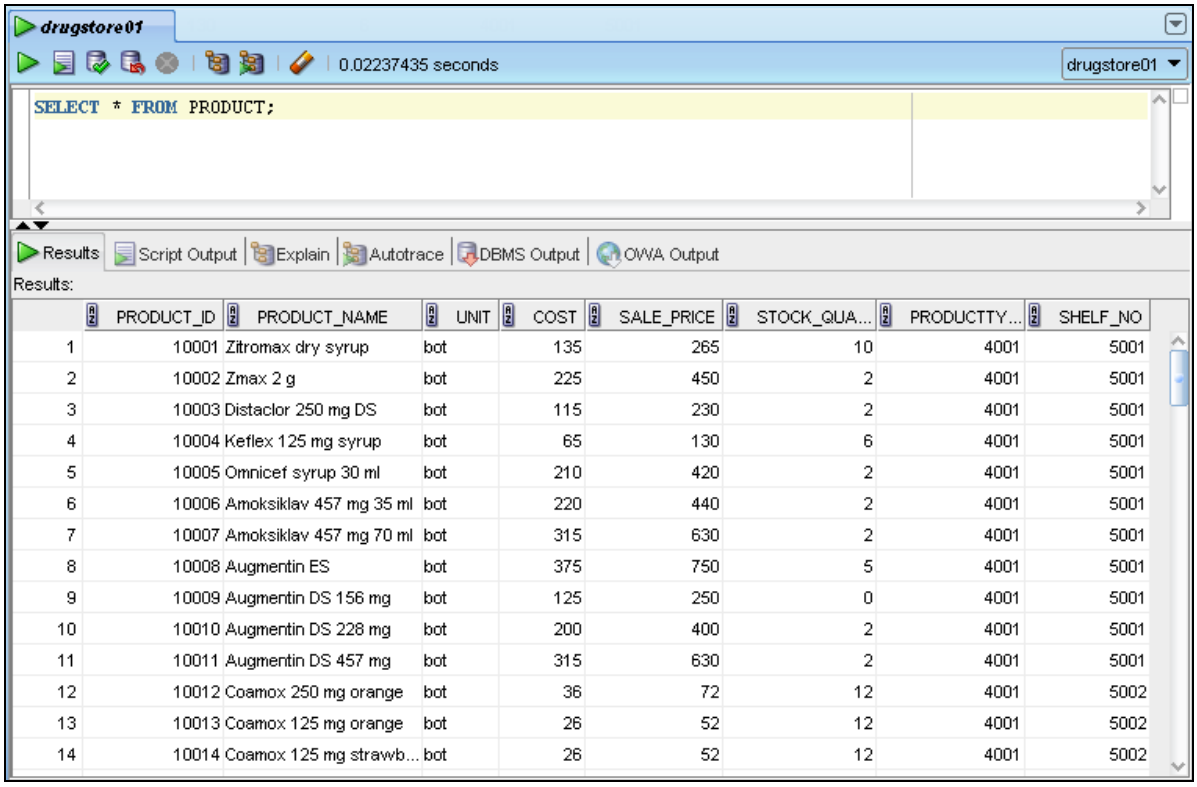
- ใช้คำสั่ง INSERT INTO เพื่อเพิ่มข้อมูลในตารางสินค้า (PRODUCT) โดยใช้เครื่องมือ SQL Developer ดังรูปที่ ข.7



รูปที่ ข.7 เพิ่มข้อมูลในตารางสินค้า (PRODUCT)

จากรูปที่ ข.7 เป็นการเพิ่มข้อมูลเข้าสู่ตารางสินค้า (PRODUCT) ซึ่งต้องใส่ข้อมูลดังนี้ รหัสสินค้า (Product_ID) ชื่อสินค้า (Product_Name) หน่วยของสินค้า (Unit) ต้นทุนต่อหน่วย (Cost) ราคาขายต่อหน่วย (Sale_Price) จำนวนสินค้าในคลังสินค้า (Stock_Quantity) รหัสประเภทสินค้า (ProductType_ID) และหมายเลขชั้นวางสินค้า (Shelf_No)

หลังจากที่ใส่ข้อมูลลงในตารางเรียบร้อยแล้ว สามารถดูข้อมูลในตารางสินค้า (PRODUCT) ดังรูปที่ ข.8



	PRODUCT_ID	PRODUCT_NAME	UNIT	COST	SALE_PRICE	STOCK_QUA...	PRODUCTTY...	SHELF_NO
1	10001	Zitromax dry syrup	bot	135	265	10	4001	5001
2	10002	Zmax 2 g	bot	225	450	2	4001	5001
3	10003	Distaclor 250 mg DS	bot	115	230	2	4001	5001
4	10004	Keflex 125 mg syrup	bot	65	130	6	4001	5001
5	10005	Omnicef syrup 30 ml	bot	210	420	2	4001	5001
6	10006	Amoksiklav 457 mg 35 ml	bot	220	440	2	4001	5001
7	10007	Amoksiklav 457 mg 70 ml	bot	315	630	2	4001	5001
8	10008	Augmentin ES	bot	375	750	5	4001	5001
9	10009	Augmentin DS 156 mg	bot	125	250	0	4001	5001
10	10010	Augmentin DS 228 mg	bot	200	400	2	4001	5001
11	10011	Augmentin DS 457 mg	bot	315	630	2	4001	5001
12	10012	Coamox 250 mg orange	bot	36	72	12	4001	5002
13	10013	Coamox 125 mg orange	bot	26	52	12	4001	5002
14	10014	Coamox 125 mg strawb...	bot	26	52	12	4001	5002

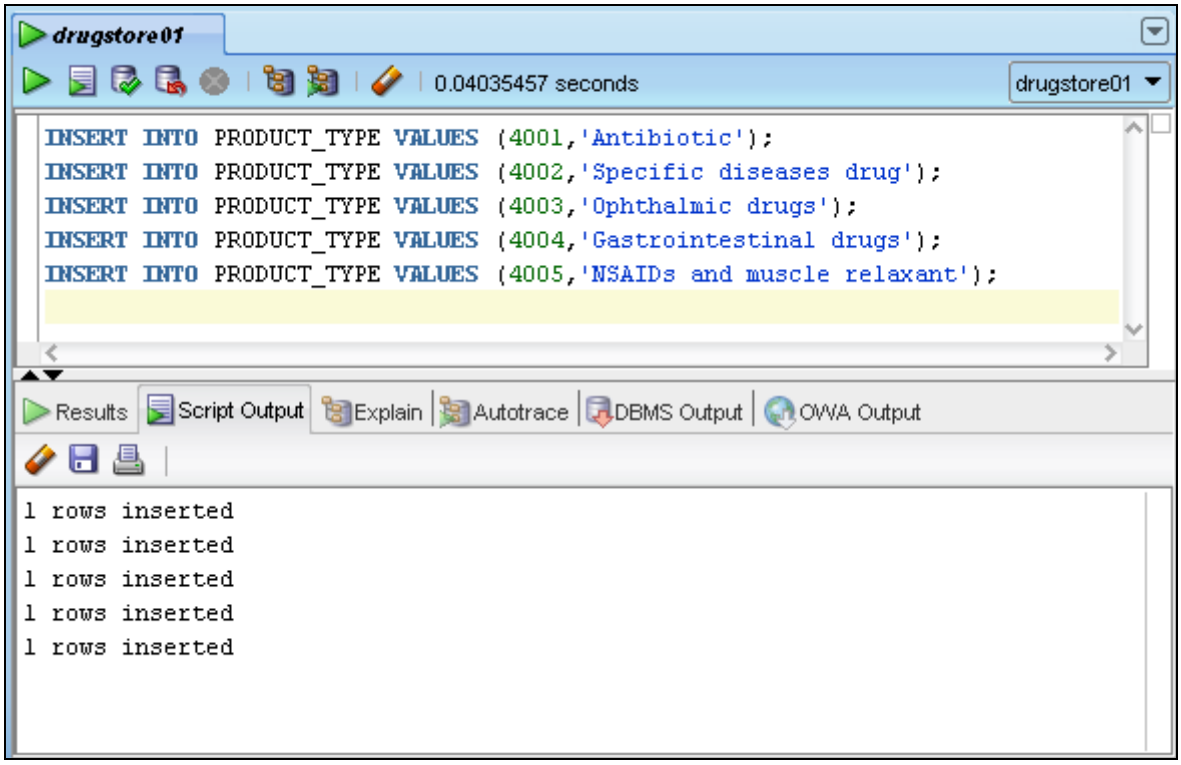
รูปที่ ข.8 ข้อมูลที่เพิ่มในตารางสินค้า (PRODUCT)

- **การเพิ่มข้อมูลในตารางประเภทสินค้า (PRODUCT_TYPE)**

- การใช้คำสั่ง INSERT INTO เพื่อเพิ่มข้อมูลในตารางประเภทสินค้า (PRODUCT_TYPE) ซึ่งมี SQL statement ดังนี้

```
INSERT INTO PRODUCT_TYPE VALUES (4001,'Antibiotic');
INSERT INTO PRODUCT_TYPE VALUES (4002,'Specific diseases drug');
INSERT INTO PRODUCT_TYPE VALUES (4003,'Ophthalmic drugs');
```

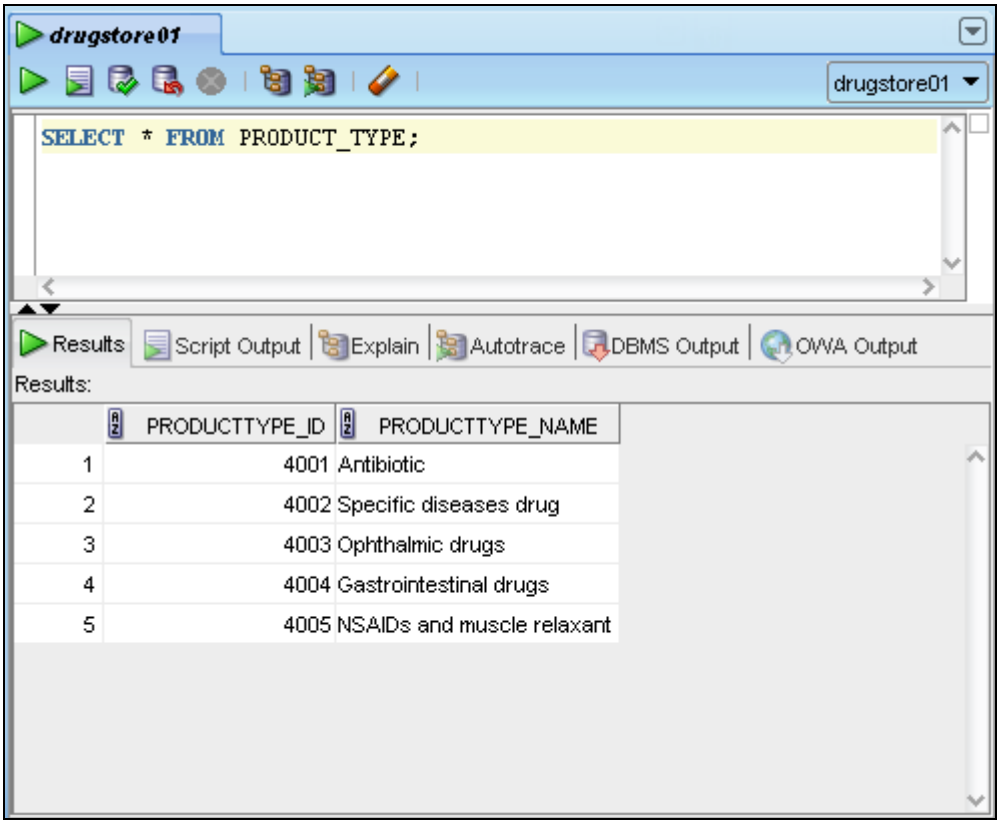
- ใช้คำสั่ง INSERT INTO เพื่อเพิ่มข้อมูลในตารางประเภทสินค้า (PRODUCT_TYPE) โดยใช้เครื่องมือ SQL Developer ดังรูปที่ ข.9



รูปที่ ข.9 เพิ่มข้อมูลในตารางประเภทสินค้า (PRODUCT_TYPE)

จากรูปที่ ข.9 เป็นการเพิ่มข้อมูลเข้าสู่ตารางประเภทสินค้า (PRODUCT_TYPE) ซึ่งต้องใส่ข้อมูลดังนี้ รหัสประเภทสินค้า (ProductType_ID) และชื่อประเภทสินค้า (ProductType_Name)

หลังจากที่ใส่ข้อมูลลงในตารางเรียบร้อยแล้ว สามารถดูข้อมูลในตารางประเภทสินค้า (PRODUCT_TYPE) ดังรูปที่ ข.10



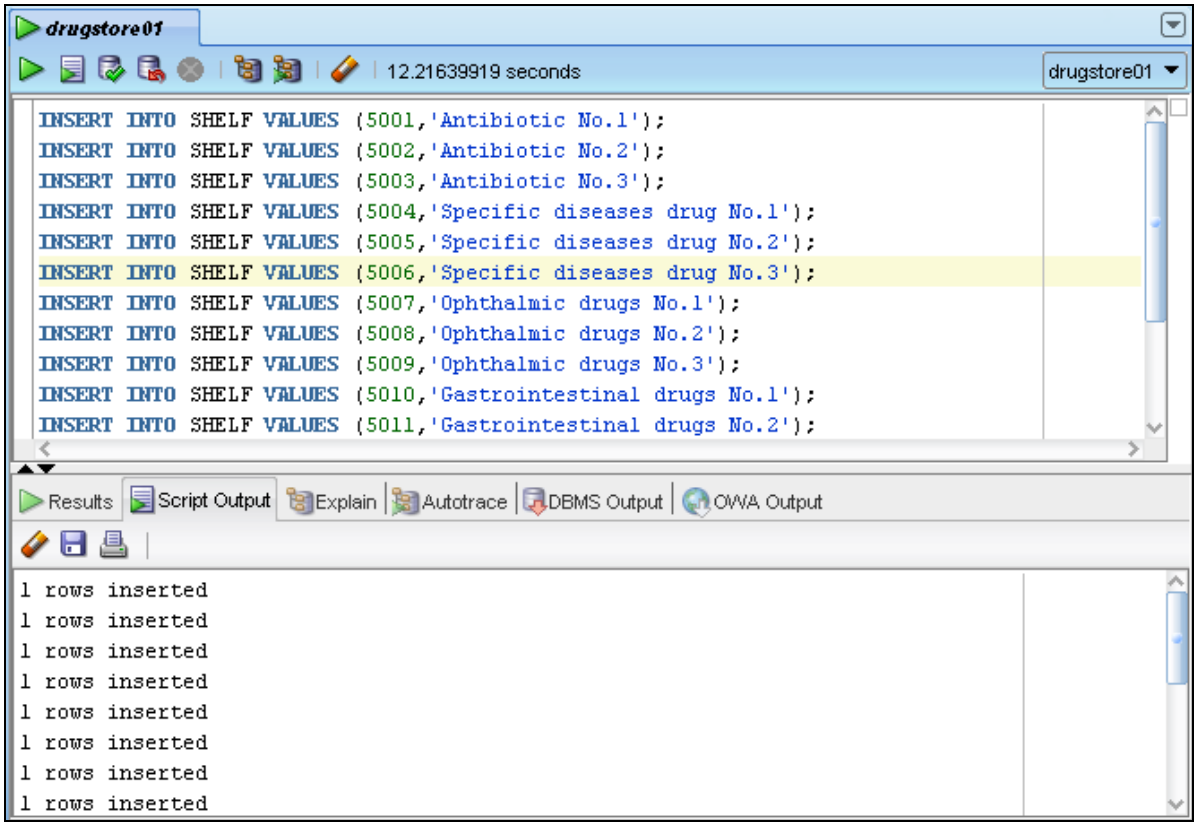
รูปที่ ข.10 ข้อมูลที่เพิ่มในตารางประเภทสินค้า (PRODUCT_TYPE)

- **การเพิ่มข้อมูลในตารางชั้นวางสินค้า (SHELF)**

- การใช้คำสั่ง INSERT INTO เพื่อเพิ่มข้อมูลในตารางชั้นวางสินค้า (SHELF) ซึ่งมี SQL statement ดังนี้

```
INSERT INTO SHELF VALUES (5001,'Antibiotic No.1');
INSERT INTO SHELF VALUES (5002,'Antibiotic No.2');
INSERT INTO SHELF VALUES (5003,'Antibiotic No.3');
```

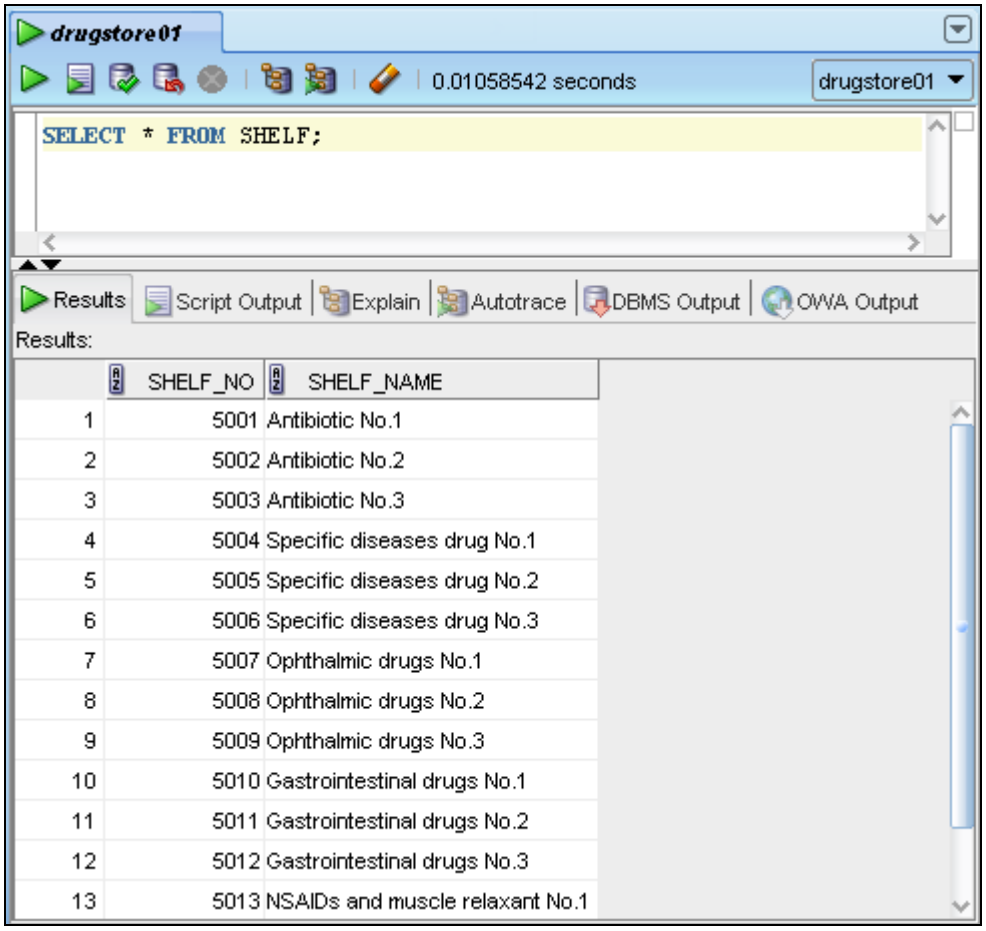
- ใช้คำสั่ง INSERT INTO เพื่อเพิ่มข้อมูลในตารางชั้นวางสินค้า (SHELF) โดยใช้เครื่องมือ SQL Developer ดังรูปที่ ข.11



รูปที่ ข.11 เพิ่มข้อมูลในตารางชั้นวางสินค้า (SHELF)

จากรูปที่ ข.11 เป็นการเพิ่มข้อมูลเข้าสู่ตารางชั้นวางสินค้า (SHELF) ซึ่งต้องใส่ข้อมูลดังนี้ หมายเลขชั้นวางสินค้า (Shelf_No) และชื่อชั้นวางสินค้า (Shelf_Name)

หลังจากที่ใส่ข้อมูลลงในตารางเรียบร้อยแล้ว สามารถดูข้อมูลในตารางชั้นวางสินค้า (SHELF) ดังรูปที่ ข.12



	SHELF_NO	SHELF_NAME
1	5001	Antibiotic No.1
2	5002	Antibiotic No.2
3	5003	Antibiotic No.3
4	5004	Specific diseases drug No.1
5	5005	Specific diseases drug No.2
6	5006	Specific diseases drug No.3
7	5007	Ophthalmic drugs No.1
8	5008	Ophthalmic drugs No.2
9	5009	Ophthalmic drugs No.3
10	5010	Gastrointestinal drugs No.1
11	5011	Gastrointestinal drugs No.2
12	5012	Gastrointestinal drugs No.3
13	5013	NSAIDs and muscle relaxant No.1

รูปที่ ข.12 ข้อมูลที่เพิ่มในตารางชั้นวางสินค้า (SHELF)

ประวัติผู้จัดทำ

ชื่อ-สกุล นางสาวเมตตา เทียนชนะไชยา

วัน เดือน ปีเกิด 6 พฤษภาคม 2530

ประวัติการศึกษา

ระดับมัธยมศึกษา มัธยมศึกษาตอนปลาย
โรงเรียนสตรีวัดระฆัง พ.ศ. 2548

ระดับปริญญาตรี วิศวกรรมศาสตรบัณฑิต สาขาวิชาวิศวกรรมอิเล็กทรอนิกส์และ
โทรคมนาคม มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าธนบุรี พ.ศ. 2552

ระดับปริญญาโท วิทยาศาสตรมหาบัณฑิต สาขาวิชาเทคโนโลยีสารสนเทศ
มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าธนบุรี พ.ศ. 2556

ประวัติการทำงาน วิศวกรระบบโปรแกรม (Application Engineer)
บริษัท บอมบาร์ดิเอร์ ทรานสปอร์เทชั่น ซิกแนล จำกัด พ.ศ. 2552 – ปัจจุบัน