BỘ GIÁO DỤC ĐÀO TẠO

TRƯỜNG ĐẠI HỌC THĂNG LONG

DỰ ÁN XÂY DỰNG KHO DỮ LIỆU CHO CPI CARD GROUP

GIÁO VIÊN HƯỚNG DẪN: **Đinh Thủy Tiên**

SINH VIÊN THỰC HIỆN: **A41538 –HOÀNG VŨ THẢO NHIÊN**

BỘ MÔN: **HỆ THỐNG THÔNG TIN**

HÀ NỘI – 2024

MỤC LỤC

CHƯƠNG 1. TÌM HIỂU VỀ TẬP ĐOÀN CPI CARD 1

1.1. Tập đoàn Thẻ CPI: Chuyên gia hàng đầu về giải pháp thanh toán an toàn . 1

CHƯƠNG 2. CHUỖI HOẠT ĐỘNG CỦA TẬP ĐOÀN CPI GROUP 3

2.1. Luồng công việc xử lý thẻ của Tập đoàn Thẻ CPI 3

2.2. Sáng kiến BI (Business Intelligence - Trí tuệ Kinh doanh) Tự phục vụ 4

2.3. Mục tiêu kinh doanh của CPI CARD 5

2.4. Vấn đề về kỹ thuật (Technical Issues) 5

CHƯƠNG 3. THIẾT KẾ KHO DỮ LIỆU CHO CPI CARD GROUP 6

3.1. Khái quát giải pháp và tổng quan về kho dữ liệu 6

3.2. Thiết kế cơ sở dữ liệu ERP 7

3.3. Tổng quan các bảng ở ERP database 9

3.4. Thiết kế kho dữ liệu cho CPI CARD 10

3.5. Các lược đồ của một số bảng 13

3.5.1. Job Fact Schema 13

3.5.2. Job shipment schema 14

3.5.3. SubJob schema 14

3.5.4. SalesLead Schema 15

3.5.5. Invoice Schema 15

3.5.6. Cost financial summary schema 16

CHƯƠNG 4. NHỮNG YÊU CẦU ĐỐI VỚI CÁC TRUY VẤN PHÂN TÍCH 17

4.1. Nhu cầu Trí tuệ Kinh doanh 17

4.2. Yêu cầu thứ hai đối với bảng truy vấn và phân tích dữ liệu 17

4.3. Yêu cầu thứ ba đối với bảng truy vấn và phân tích dữ liệu 17

4.4. Các loại truy vấn 18

4.5. Các Vấn đề về Mô hình Dữ liệu trong Kho dữ liệu Tập đoàn Thẻ CPI 18

4.5.1. Những lưu ý về Thiết kế Bảng 18

CHƯƠNG 5. PHÂN TÍCH DỮ LIỆU CHO CPT CARD.....20

5.1. Xây dựng các truy vấn cơ sở 20

5.1.1. Base query 1.....20

5.1.2. Base query 2.....21

5.1.3. Base query 3.....22

5.1.4. Base query 4.....23

5.1.5. Base query 5.....24

5.1.6. Base query 6.....25

5.2. Xây dựng các truy vấn phân tích.....27

5.2.1. Truy vấn phân tích liên quan đến xu hướng doanh thu của khách hàng...27

5.2.2. Truy vấn phân tích liên quan đến tóm tắt doanh thu và chi phí.....28

5.2.3. Truy vấn phân tích liên quan đến trả về.....33

5.2.4. Truy vấn phân tích liên quan đến sự chậm trễ trong hợp đồng 34

CHƯƠNG 6. TRỰC QUAN HÓA DỮ LIỆU CHO CPI CARD.....37

DANH MỤC HÌNH ẢNH

Bảng 3.1. Thiết kế các bảng dimensions và fact 11
Bảng 3.2. Các thước đo, nguồn dữ liệu liên quan và các thuộc tính tổng hợp thước đo. 12
Bảng 3.3. Phân loại các bảng vào các Cubes 13

Hình 2.1. Simplified workflow diagram for Card processing 3
Hình 3.1. Nguồn dữ liệu cho kho dữ liệu 7
Hình 3.2. ERD cho tập hợp con của ERP database 8
Hình 3.3. Tổng quan các bảng ở ERP database 9
Hình 3.4. Job Fact Schema 13
Hình 3.5. Job Shipment Schema 14
Hình 3.6. Sub Job Schema 14
Hình 3.7. Sales Lead Schema 15
Hình 3.8. Invoice Schema 15
Hình 3.9. Cost financial summary schema 16

CHƯƠNG 1. TÌM HIỂU VỀ TẬP ĐOÀN CPI CARD

Phần này em sẽ giới thiệu tổng quan về tập đoàn CPI CARD

1.1. Tập đoàn Thẻ CPI: Chuyên gia hàng đầu về giải pháp thanh toán an toàn

- **Hơn 30 năm kinh nghiệm**

Tập đoàn Thẻ CPI tự hào có di sản 30 năm đáng chú ý trong ngành thẻ thanh toán. Họ đã khẳng định vị thế dẫn đầu toàn cầu, chế tạo thẻ tài chính cho các thương hiệu nổi tiếng như Visa, MasterCard, American Express, Discover và Interac. Khách hàng của họ bao gồm các tổ chức tài chính, nhà cung cấp dịch vụ, bộ xử lý thẻ và chương trình thẻ ghi nợ trả trước - cả nội địa (Hoa Kỳ) và quốc tế.

- **Vượt xa thẻ truyền thống: Một loạt các giải pháp an toàn**

Tập đoàn Thẻ CPI không chỉ cung cấp thẻ ghi nợ, thẻ tín dụng và thẻ trả trước tiêu chuẩn. Họ cung cấp nhiều loại thẻ nhựa cho người tiêu dùng, bao gồm cả thẻ nhận dạng. Ngoài ra, họ đáp ứng nhu cầu ngày càng tăng về thẻ thông minh không tiếp xúc, cung cấp các giải pháp sáng tạo như thẻ RFID, thẻ kiểm soát truy cập và thẻ chip EMV (Europay, MasterCard và Visa). Họ thậm chí còn sản xuất thẻ MicroSD cho các tổ chức tư nhân và công cộng, thể hiện cam kết của họ về lưu trữ dữ liệu an toàn.

Dịch vụ của họ mở rộng sang đóng gói thẻ quà tặng, cho phép tùy chỉnh hoàn toàn. Hãy tưởng tượng bạn có thể thiết kế và sản xuất bất kỳ loại bao bì thẻ quà tặng độc đáo nào, thêm một nét cá nhân! Họ cung cấp dịch vụ xử lý chương trình thẻ an toàn và không an toàn, xử lý các hoạt động hàng ngày cho các doanh nghiệp ở mọi quy mô. Hơn nữa, họ cung cấp dịch vụ phát hành, phân phối và giải pháp thẻ thông minh cho doanh nghiệp và khách hàng.

- **Cơ sở hạ tầng an ninh chưa từng có**

Tập đoàn Thẻ CPI sở hữu mạng lưới sản xuất an toàn rộng lớn nhất ở Bắc Mỹ, có trụ sở chính tại Colorado, Hoa Kỳ. Họ có một cơ sở an toàn chuyên dụng ở Canada để hỗ trợ sản xuất thẻ hàng ngày và quy mô lớn. Cơ sở hạ tầng mạnh mẽ này cho phép họ xử lý các chương trình thẻ thanh toán đa dạng, từ các sáng kiến tiếp thị cá nhân đến triển khai thẻ quy mô lớn. Phạm vi tiếp cận của họ vượt ra ngoài biên giới, với các địa điểm trải dài khắp Hoa Kỳ, Canada và Vương quốc Anh.

- **Đối tác đáng tin cậy cho hơn 3.000 khách hàng**

Tập đoàn Thẻ CPI phục vụ cho một nhóm khách hàng đa dạng hơn 3.000 người, cả trực tiếp và gián tiếp. Danh sách ấn tượng của họ bao gồm các nhà phát hành thẻ ghi nợ và thẻ tín dụng lớn, nhà cung cấp thẻ trả trước, ngân hàng cộng đồng độc lập, hợp tác xã tín dụng, bộ xử lý thẻ và nhà cung cấp dịch vụ.

- **Cách mạng EMV: Dẫn đầu**

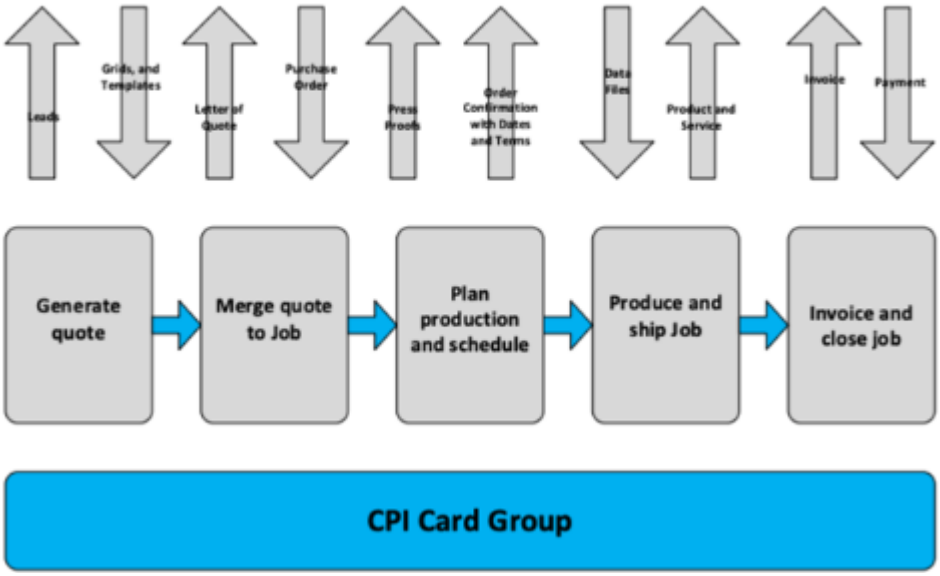
Việc vi phạm bảo mật cao cấp gần đây đã nhấn mạnh sự cấp bách để thị trường Hoa Kỳ áp dụng tiêu chuẩn EMV. Hệ thống sáng tạo này, do Europay, MasterCard và Visa tạo ra, sử dụng công nghệ chip thông minh để nâng cao bảo mật và phê duyệt giao dịch cho thẻ thanh toán, thiết bị đầu cuối và ATM.

Thẻ EMV lưu trữ dữ liệu trên chip bảo mật thay vì dải từ dễ bị tổn thương, mặc dù một số thẻ có thể giữ lại dải từ để tương thích. Ngoài ra, thẻ EMV tận dụng xác minh chip và mã PIN hoặc chip và chữ ký để tăng cường bảo mật. Đầu đọc cho những thẻ này có thể sử dụng kết nối vật lý hoặc công nghệ không dây qua RFID.

Mặc dù việc áp dụng thẻ thông minh ban đầu chậm chạp ở Hoa Kỳ, nhưng một sự thay đổi đáng kể đang diễn ra với hầu hết các ngân hàng và tổ chức tài chính chuyển sang sử dụng thẻ hỗ trợ EMV. CPI, là nhà sản xuất thẻ thông minh lớn nhất ở Bắc Mỹ và là người tiên phong trong việc cung cấp thẻ tín dụng hỗ trợ chip trong khu vực, được định vị tốt để dẫn đầu sự thay đổi này. Họ tích cực hỗ trợ khách hàng triển khai cơ sở hạ tầng thanh toán EMV, củng cố bảo mật doanh nghiệp của họ. Sản xuất của CPI đã chứng kiến sự bùng nổ, sản xuất hàng triệu thẻ EMV chỉ trong năm ngoái. Họ cũng tích cực tham gia vào các chương trình thí điểm với các ngân hàng lớn của Hoa Kỳ. Cam kết của họ vượt xa việc sản xuất, vì họ tích cực mở rộng mạng lưới và quan hệ đối tác, củng cố vị thế là nhà quản lý dịch vụ đáng tin cậy cho các ứng dụng thanh toán di động

CHƯƠNG 2. CHUỖI HOẠT ĐỘNG CỦA TẬP ĐOÀN CPI GROUP

2.1. Luồng công việc xử lý thẻ của Tập đoàn Thẻ CPI



Hình 2.1. Simplified workflow diagram for Card processing

1. **Tạo Khách hàng tiềm năng và Báo giá:**
   - Nhân viên bán hàng liên hệ với khách hàng tiềm năng.
   - Đối với mỗi khách hàng quan tâm, nhân viên tạo báo giá bán hàng riêng vì mỗi sản phẩm sản xuất đều có đặc điểm riêng.
   - Báo giá bao gồm vật liệu, nhân công, máy móc và các chi phí khác để tạo ra sản phẩm cuối cùng.
   - Trong hầu hết các trường hợp, khách hàng nhận được thư báo giá cùng với báo giá chi tiết.

2. **Đơn đặt hàng (PO):**
   - Khi khách hàng đồng ý với báo giá, họ sẽ gửi đơn đặt hàng (PO) cho công ty.
   - Lúc này, một công việc (job) được tạo trong hệ thống ERP và báo giá được tạo bên ngoài hệ thống ERP được hợp nhất vào công việc, bao gồm tất cả dữ liệu cần thiết để xây dựng sản phẩm cuối cùng.

3. **Kế hoạch Sản xuất và Duyệt mẫu:**
   - Sau khi bộ phận dịch vụ khách hàng nhập công việc, bộ phận lập kế hoạch sản xuất sẽ xem xét báo giá bán hàng cần thiết để thực hiện công việc.
   - Công việc của bộ phận lập kế hoạch sản xuất là chuyển đổi báo giá bán hàng và dữ liệu công việc thành kế hoạch sản xuất thực tế để tạo ra sản phẩm cuối cùng.

- Trong quá trình này, mẫu sản phẩm cũng được tạo ra để khách hàng duyệt.

4. **Lên lịch sản xuất:**
   - Sau khi mẫu được phê duyệt, nhóm lập kế hoạch sản xuất lên lịch công việc trong hệ thống hoạch định tích hợp, hệ thống này chia sẻ thông tin với hệ thống ERP.
   - Lúc này, sản phẩm chạy qua quy trình sản xuất.
5. **Sản xuất, Kiểm tra và Kiểm toán:**
   - Trong suốt quá trình sản xuất, dữ liệu sản xuất được nhập vào hệ thống ERP để thu thập số liệu sản xuất.
   - Hệ thống ERP cũng quản lý và thực hiện các cuộc kiểm toán trong suốt quá trình sản xuất.
6. **Giao hàng và Xuất hóa đơn:**
   - Sau khi sản phẩm trong một công việc được tạo ra và vượt qua tất cả các kiểm tra chất lượng và kiểm toán, sản phẩm được giao cho khách hàng.
   - Sau khi vận chuyển sản phẩm, hóa đơn được tạo và gửi cho khách hàng. Đối với các công việc lớn, nhiều lần vận chuyển và hóa đơn có thể cần thiết.
   - Sau khi công việc được vận chuyển và xuất hóa đơn, công ty có thể ghi nhận doanh thu cho toàn bộ hoặc một phần công việc đã được sản xuất.

2.2. Sáng kiến BI (Business Intelligence - Trí tuệ Kinh doanh) Tự phục vụ

CPI Card Group có một sáng kiến mới về trí tuệ kinh doanh với báo cáo tự phục vụ. CPI muốn báo cáo hướng đến việc ra quyết định cấp cao hơn cho trí tuệ kinh doanh, không chỉ các quyết định vận hành. Báo cáo tự phục vụ là một cách tiếp cận cho phép người dùng doanh nghiệp truy cập và làm việc với dữ liệu của công ty mà không cần sự tham gia của nhân viên công nghệ thông tin.

- **Các yếu tố thúc đẩy lớn nhất cho sáng kiến này là:**
  + **Thời gian quay vòng báo cáo cho doanh nghiệp:** Các báo cáo liên quan đến công việc của nhân viên công nghệ thông tin do cấu trúc phức tạp của cơ sở dữ liệu.
  + **B backlog (số lượng tồn đọng) báo cáo được yêu cầu vượt quá vài tháng** không cho phép đưa ra các quyết định kinh doanh nhanh chóng và chính xác.
  + **Chi phí phát triển cho các nhu cầu báo cáo động:** Ba nhân viên toàn thời gian được dành riêng cho việc tạo báo cáo mới. Tuy nhiên, nhân viên này không đủ để hỗ trợ cho nhu cầu báo cáo ngày càng tăng khi doanh nghiệp mở rộng, đặc biệt là với các vụ mua bán.
  + **Thiếu báo cáo hợp nhất trên các nguồn dữ liệu:** CPI Card Group cần các báo cáo kết hợp các nguồn dữ liệu được sử dụng trong các hệ thống cho hoạt động. Báo cáo hợp nhất

kết hợp dữ liệu từ quản lý quan hệ khách hàng (CRM), ERP cho sản xuất công việc và kế toán tài chính.

+ **Không có tính năng tự phục vụ cho người dùng cuối để tạo báo cáo:** CPI Card Group muốn trao quyền cho các nhà phân tích kinh doanh của mình bằng các công cụ báo cáo. Các nhà phân tích kinh doanh

2.3. Mục tiêu kinh doanh của CPI CARD

- Theo dõi và so sánh doanh thu và chi phí: Công ty cần theo dõi doanh thu và chi phí theo các chiều chính như địa điểm, loại hình doanh thu, đại lý bán hàng và khách hàng qua các khoảng thời gian khác nhau. Điều này giúp họ hiểu rõ hơn về hiệu suất kinh doanh và nhận diện các khu vực cần cải thiện.
- Quản lý chất lượng sản phẩm và hợp đồng: Giám sát chất lượng sản phẩm và hiệu suất hợp đồng là một phần quan trọng trong việc duy trì sự hài lòng của khách hàng và tối ưu hóa quy trình sản xuất. Công ty cần theo dõi các ngày giao hàng, số lượng sản phẩm bị trả lại và các vấn đề chất lượng khác để đảm bảo sản phẩm đáp ứng các tiêu chuẩn cao nhất.
- Phân tích tài chính và đối chiếu: Công ty cần theo dõi và đánh giá hiệu suất tài chính thông qua các truy vấn phân tích tài chính, bao gồm dự báo doanh thu và lập ngân sách, cũng như đối chiếu các báo cáo tài chính để loại bỏ các giao dịch nội bộ và chi phí trùng lặp.

2.4. Vấn đề về kỹ thuật (Technical Issues)

Vấn đề kỹ thuật mà CPI Card Group gặp phải bao gồm:

- Thiếu chi tiết yêu cầu truy vấn phân tích: CPI Card Group chưa cung cấp được các chi tiết cụ thể về yêu cầu truy vấn phân tích, làm cho việc xác định các chỉ số cần theo dõi trở nên khó khăn và thiếu chính xác.
- Thiếu sự tự tin và kỹ năng phân tích: Không có cuộc phỏng vấn điều hành nào được thực hiện để làm rõ yêu cầu truy vấn phân tích, dẫn đến việc thiếu sự tự tin và kỹ năng cần thiết trong việc thiết kế và thực hiện các truy vấn phân tích dữ liệu.
- Phát triển thông tin kinh doanh chưa trưởng thành: CPI Card Group chưa đạt được mức độ trưởng thành cần thiết trong việc phát triển hệ thống thông tin kinh doanh để có thể khai thác và sử dụng hiệu quả dữ liệu từ kho dữ liệu cho các mục đích phân tích.

CHƯƠNG 3. THIẾT KẾ KHO DỮ LIỆU CHO CPI CARD GROUP

3.1. Khái quát giải pháp và tổng quan về kho dữ liệu

Để giải quyết các vấn đề mà CPI Card Group đang gặp phải, một giải pháp toàn diện về kho dữ liệu (Data Warehouse) được đề xuất như sau:

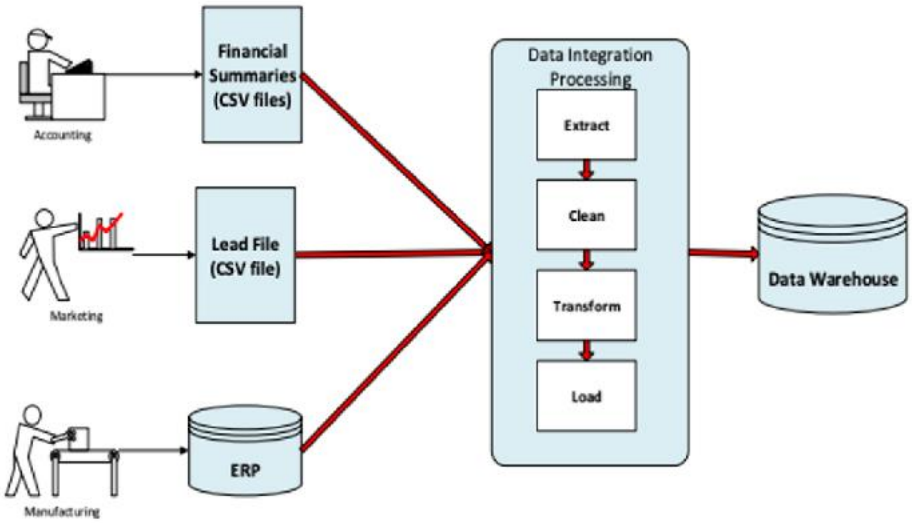
- Xây dựng kho dữ liệu tích hợp (Integrated Data Warehouse):
  + Tích hợp dữ liệu: Tạo ra một kho dữ liệu tập trung từ các hệ thống khác nhau, bao gồm dữ liệu bán hàng, sản xuất, tài chính và dịch vụ khách hàng. Sử dụng công cụ ETL (Extract, Transform, Load) để tích hợp dữ liệu từ các nguồn khác nhau vào một nền tảng duy nhất.
  + Cập nhật thường xuyên: Thiết lập quy trình cập nhật dữ liệu thường xuyên để đảm bảo kho dữ liệu luôn chứa thông tin mới nhất và chính xác nhất.
- Hỗ trợ truy vấn phân tích (Analytical Query Support):
  + Phát triển truy vấn phân tích: Thiết kế các truy vấn phân tích chi tiết để theo dõi các chỉ số kinh doanh quan trọng như doanh thu, chi phí, tỷ lệ thành công của lead và chất lượng sản phẩm. Sử dụng các công cụ BI (Business Intelligence) để tạo báo cáo và dashboard dễ hiểu.
  + Đào tạo nhân viên: Tổ chức các khóa đào tạo cho nhân viên về cách sử dụng các công cụ BI và hiểu rõ các truy vấn phân tích để họ có thể khai thác tối đa dữ liệu.
- Tăng Cường Kiểm Soát Chất Lượng (Quality Control Enhancement):
  + Giám sát chất lượng: Thiết lập các truy vấn để giám sát chất lượng sản phẩm, bao gồm số lượng sản phẩm bị trả lại, sự sai lệch về ngày giao hàng, và các vấn đề khác liên quan đến chất lượng.
  + Báo cáo chất lượng: Tạo các báo cáo định kỳ về chất lượng sản phẩm và hiệu suất hợp đồng, giúp công ty nhanh chóng nhận diện và xử lý các vấn đề chất lượng.
- Phân tích tài chính (Financial Analysis):
  + Giám sát dự báo và ngân sách: Phát triển các truy vấn để theo dõi hiệu suất dự báo doanh thu và lập ngân sách, so sánh giữa thực tế và dự báo để xác định độ chính xác.
  + Đối chiếu tài chính: Thiết lập các quy trình đối chiếu để loại bỏ các giao dịch nội bộ và chi phí trùng lặp, đảm bảo dữ liệu tài chính chính xác và minh bạch.
- Triển khai công nghệ hiện đại (Implementing Modern Technology):
  + Sử dụng Big Data và AI: Áp dụng công nghệ Big Data và AI để phân tích dữ liệu lớn và phức tạp, cung cấp các phân tích dự đoán và ra quyết định thông minh.
  + Công Cụ Visualization: Sử dụng các công cụ trực quan hóa dữ liệu như Tableau, Power BI để tạo các báo cáo trực quan, dễ hiểu và dễ dàng theo dõi các chỉ số kinh doanh quan trọng.

- Thiết Lập Quy Trình và Chính Sách (Establishing Processes and Policies):
  + Chính sách dữ liệu: Xây dựng chính sách về quản lý dữ liệu, bảo mật dữ liệu và quyền truy cập, đảm bảo dữ liệu được bảo vệ và sử dụng đúng mục đích.
  + Quy trình xử lý dữ liệu: Thiết lập quy trình chuẩn cho việc xử lý và phân tích dữ liệu, đảm bảo tính nhất quán và chất lượng của dữ liệu.

Bằng cách triển khai các giải pháp trên, CPI Card Group có thể nâng cao khả năng phân tích dữ liệu và cải thiện hiệu suất kinh doanh, từ đó đạt được các mục tiêu chiến lược của mình.

Kho dữ liệu sử dụng ba nguồn dữ liệu như được mô tả như trong hình bên dưới.

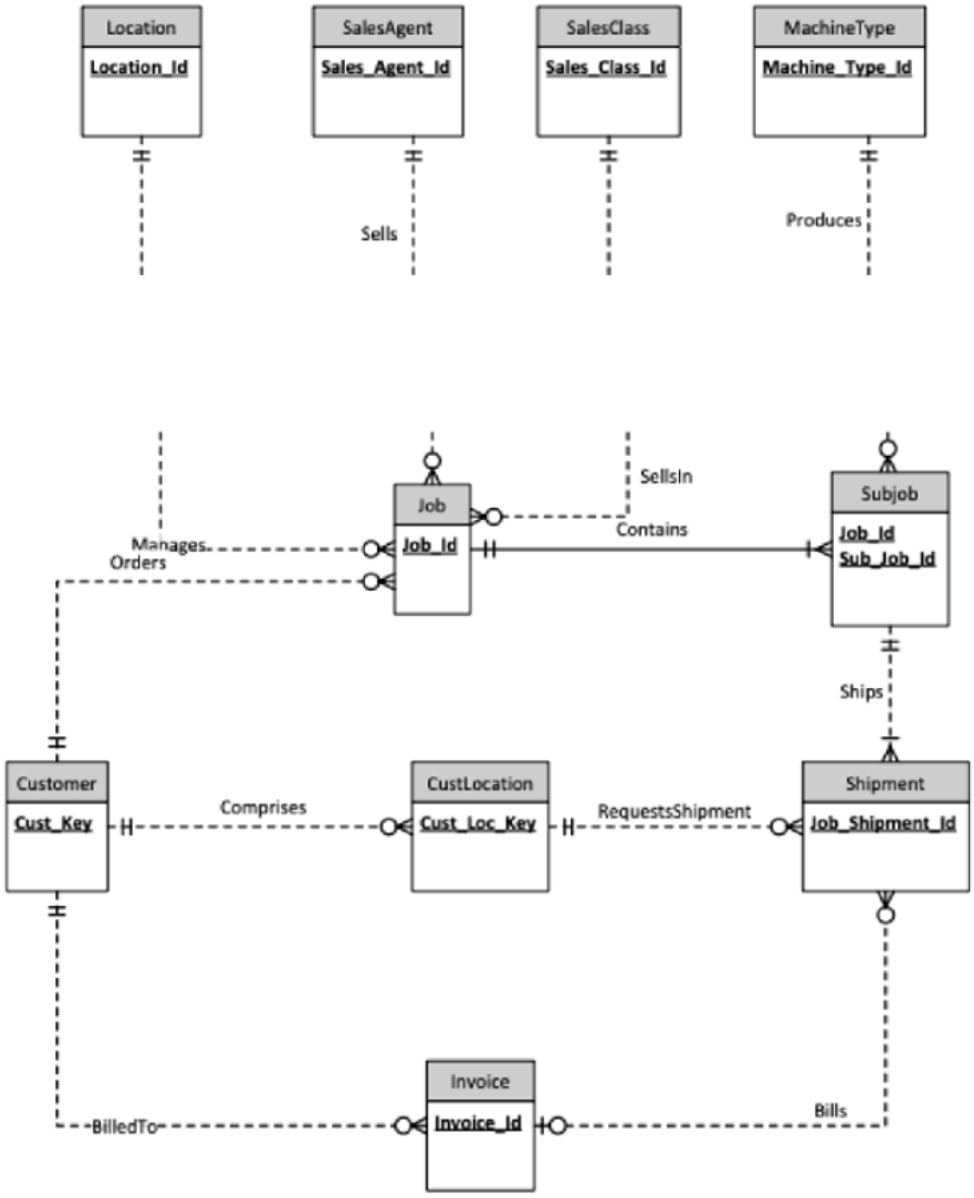
- Cơ sở dữ liệu ERP là nguồn dữ liệu chính được ngành sản xuất sử dụng để quản lý công việc, công việc phụ, lô hàng và hóa đơn.
- Tệp chính và tóm tắt tài chính là nguồn dữ liệu thứ cấp, cả ở định dạng bảng tính.
- Tệp khách hàng tiềm năng và bản tóm tắt tài chính được chuẩn bị từ các nguồn dữ liệu khác được bộ phận tiếp thị và kế toán sử dụng.



Hình 3.1. Nguồn dữ liệu cho kho dữ liệu

3.2. Thiết kế cơ sở dữ liệu ERP

Cơ sở dữ liệu ERP hỗ trợ xử lý hoàn chỉnh các công việc liên quan đến lập kế hoạch, sản xuất, vận chuyển, lập hóa đơn và xử lý thanh toán cũng như kế toán. Tuy nhiên, các chi tiết đầy đủ không quan trọng đối với trường hợp này.



Hình 3.2. ERD cho tập hợp con của ERP database

3.3. Tổng quan các bảng ở ERP database

Loại đối tượng	Mô tả
Customer	Các tổ chức yêu cầu công việc. Khách hàng tham gia vào các báo giá được ghi lại trong CRM chứ không phải trong ERP. Trong ERP, khách hàng được ghi lại trong một công việc.
CustomerLocation	Vị trí của khách hàng mà thẻ được giao đến.
Invoice	Bộ sưu tập các lô hàng được thanh toán cho khách hàng. Hóa đơn được tạo sau các lô hàng liên quan, vì vậy mối quan hệ Hóa đơn là tùy chọn.
Job	Hợp đồng cho một số lượng thẻ được tạo sau khi khách hàng chấp nhận báo giá
Location	Vị trí của công ty quản lý một công việc
MachineType	Loại máy được sử dụng để sản xuất thẻ trong một công việc phụ
SalesAgent	Nhân viên được ghi nhận là đã nhận được công việc
SalesClass	Loại sản phẩm trên một công việc
Shipment	Bộ sưu tập các thẻ được giao cho khách hàng sau khi sản xuất trong một công việc phụ
SubJob	Một phần nhỏ của công việc được sản xuất bằng loại máy. Nhận dạng phụ thuộc vào Job.

		FACT						
		Job	SubJob	JobShipment	FinSumCost	FinSumSales	Lead	Invoice
DIMENSION	Time	X(3)	X(2)	X(2)	X(2)	X(2)	X	X(2)
	Customer	X	X				X	X
	CustLoc			X				
	SalesClass	X	X	X	X	X	X	X
	SalesAgent	X					X	X
	Location	X	X	X	X	X	X	X
	MachineType		X		X			

Hình 3.3. Tổng quan các bảng ở ERP database

3.4. Thiết kế kho dữ liệu cho CPI CARD

Dimension	Attributes	Hierarchies	Data sources
Customer	Cust_Key, Cust_Name Location, E_Mail_Address, Terms_Code	Country, State, City Country, Zip Top level, second level domains	Customer table in ERP
Customer location	Cust_Loc_Key, Cust_Name Location E_Mail_Address Terms_Code	Country, State, City Country, Zip Top level, second level domains	CustomerLoc table in ERP
Location	Location_Id, Location_Name		Location table in ERP
Machine Type	Machine_Type_Id, Manufacturer, Model, Rate_Per_Hour, Number_Of_Machines		MachineType table in ERP
Sales agent	Sales_Agent_Id, Sales_Agent_Name, Location		SalesAgent table in ERP
Sales class	Sales_Class_Id, Sales_Class_Desc	State, country	SalesClass table in ERP
Time	Various dates: Date_Invoiced, Date_Due, Contract_Date, Date_Promised, Date_Ship_By. Date_Prod_Begin, Date_Prod_End, Actual_Shiip_Date, Requested_Ship_Date,	Year, quarter, month, day Year, week	Date columns in Job, Lead,Shipment, Subjob, Invoice,FinancialCostSummary, andFinancialSalesSummary

	Begin_Date, End_Date, Created_Date		

Bảng 3.1. Thiết kế các bảng dimensions và fact

Data Source	Measures	Aggregation Properties
Lead	Success, Quote_Price, Quote_Qty	Non-additive Non-additive Additive
Job	Quantity_Ordered, Quotation_Amount, Quotation_Ordered	Additive Additive Additive
Subjob	Cost_Material, Cost_Labor, Cost_Overhead, Machine_Hours, Quantity_Produced	Additive Additive Additive Additive Additive
Shipment	Actual_Quantity, Boxes, Quantity_Per_Box, Quantity_Per_Partial_Box	Additive Additive when multiplied by Quantity_Per_Box Additive
Invoice	Invoice_amount	
Financial_Sales_Summary	Actual_Units, Actual_Amount, Forecast_Units, Forecast_Amount	Additive for all

Financial_Cost_Summary	Actual_Units, Actual_Labor_Costs, Actual_Material_Cost, Actual_Overhead_Cost, Budget_Units, Budget_Material_Cost, Budget_Machine_Cost, Budget_Overhead_Cost	Additive for all

Bảng 3.2. Các thước đo, nguồn dữ liệu liên quan và các thuộc tính tổng hợp thước đo.

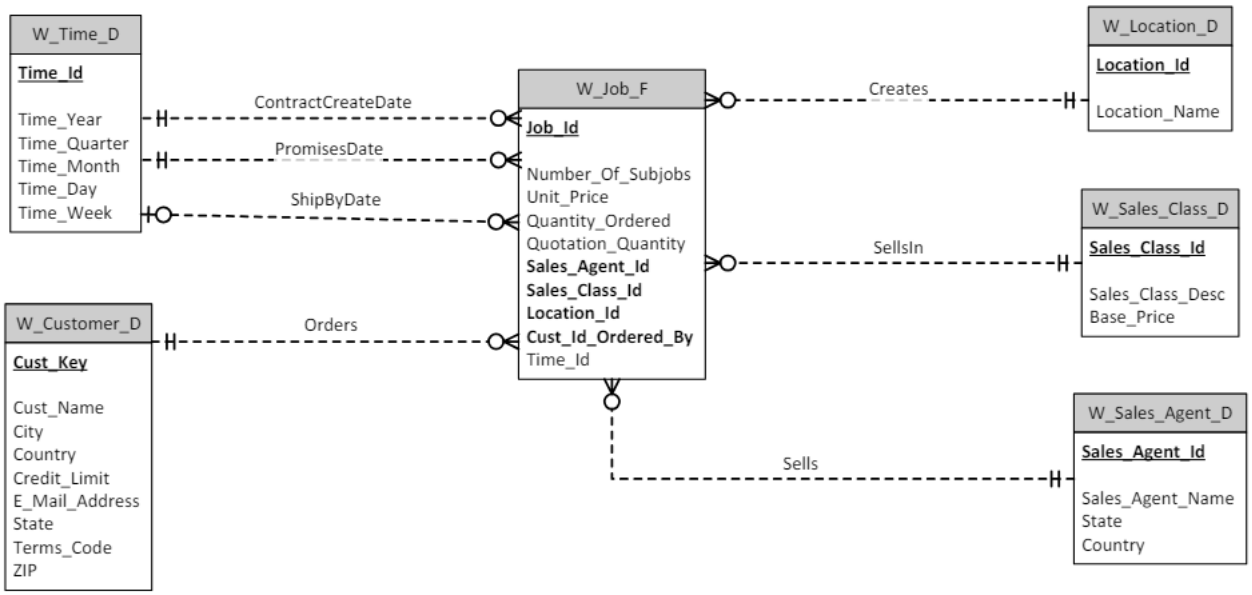
Cube	Dimensions	Measures
Lead	Customer, Sales_Agent, Sales_Class, Location, Time	Success, Quote_Price, Quote_Qty
Job	Customer, Sales_Agent, Sales_Class, Location, Time	Quantity_Ordered, Quotation_Amount, Quotation_Ordered
Subjob	Machine_Type, Customer, Sales_Class, Location, Time	Cost_Material, Cost_Labor, Cost_Overhead, Machine_Hours, Quantity_Produced
Shipment	CustLocation, Location, Sales_Class, Time	Actual_Quantity, Boxes, Quantity_Per_Box, Quantity_Per_Partial_Box
Invoice	Customer, Location,	Invoice_Amount

	Sales_Class, Sales_Agent, Time	
Financial_Sales_Summary	Sales_Class, Location, Time	Actual_Units, Actual_Amount, Forecast_Units, Forecast_Amount
Financial_Cost_Summary	Machine_Type, Sales_Class, Location, Time	Actual_Units, Actual_Labor_Costs, Actual_Material_Cost, Actual_Overhead_Cost, Budget_Units, Budget_Material_Cost, Budget_Machine_Cost, Budget_Overhead_Cost

Bảng 3.3. Phân loại các bảng vào các Cubes

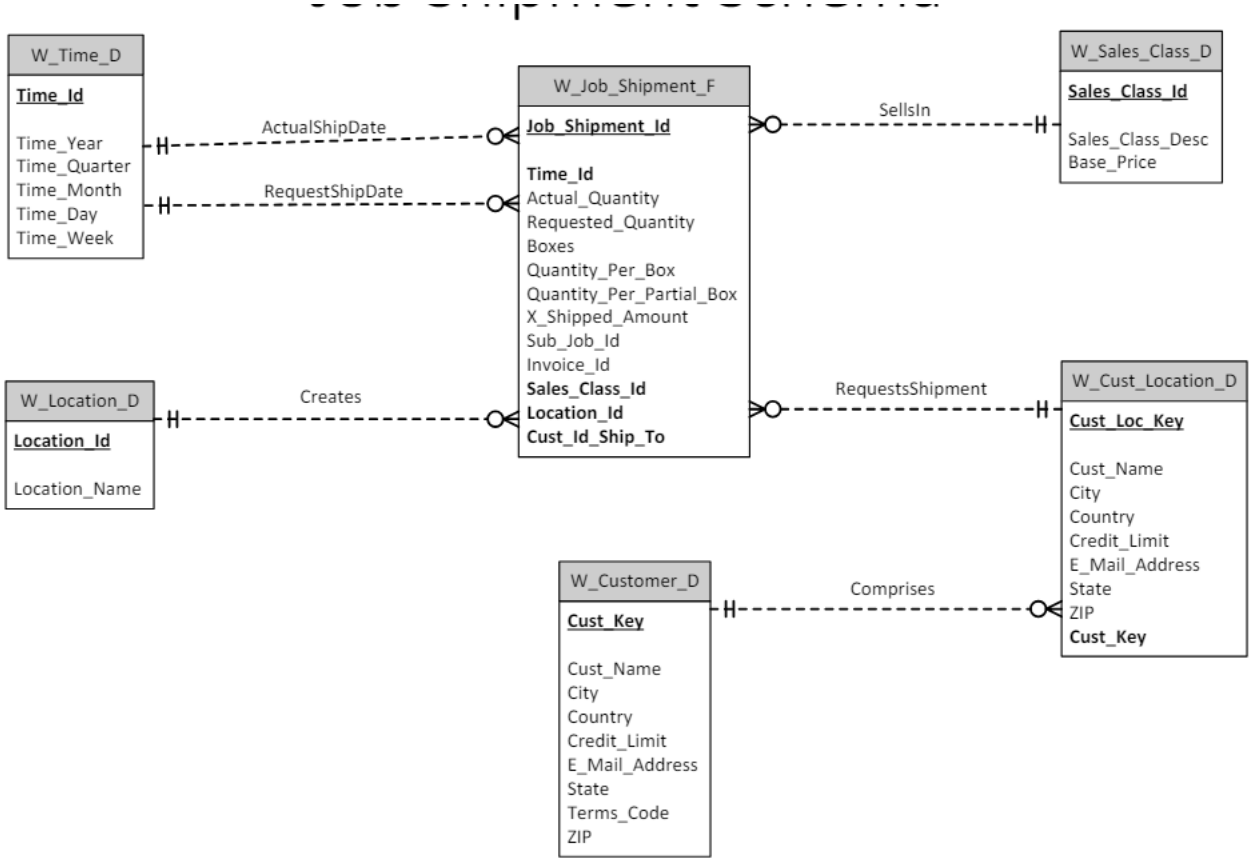
3.5. Các lược đồ của một số bảng

3.5.1. Job Fact Schema



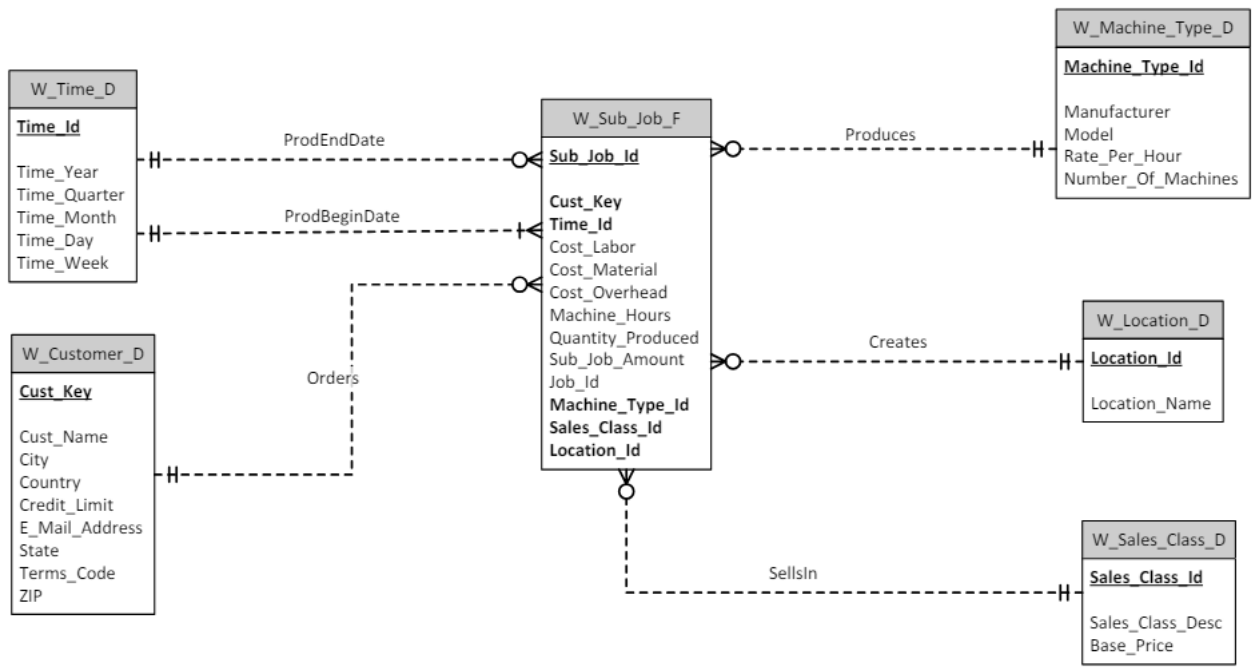
Hình 3.4. Job Fact Schema

3.5.2. Job shipment schema



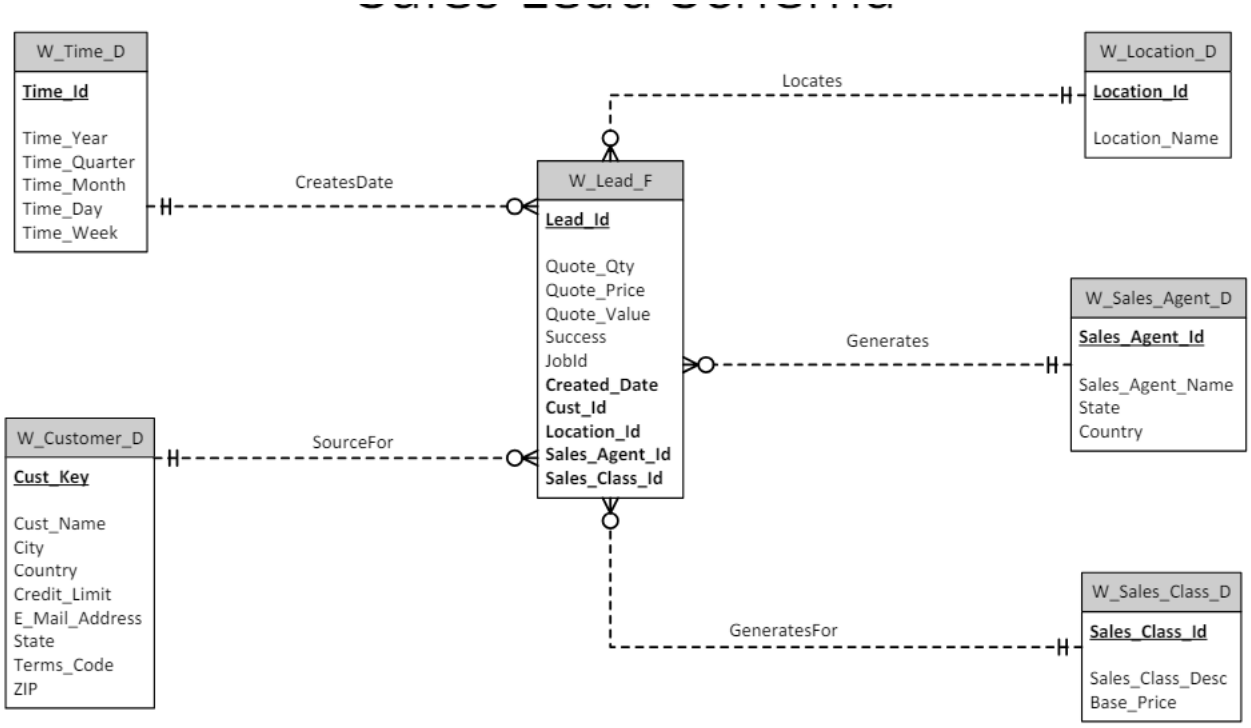
Hình 3.5. Job Shipment Schema

3.5.3. SubJob schema



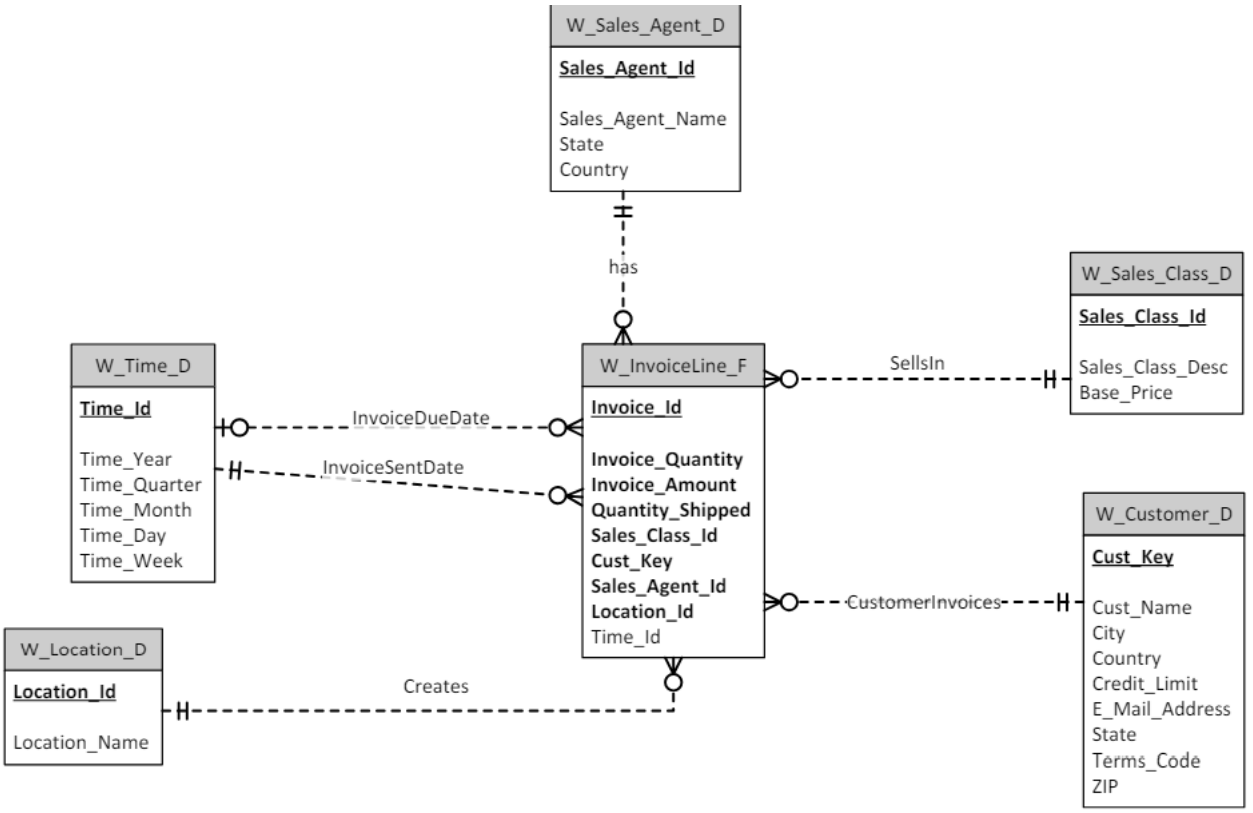
Hình 3.6. Sub Job Schema

3.5.4. SalesLead Schema



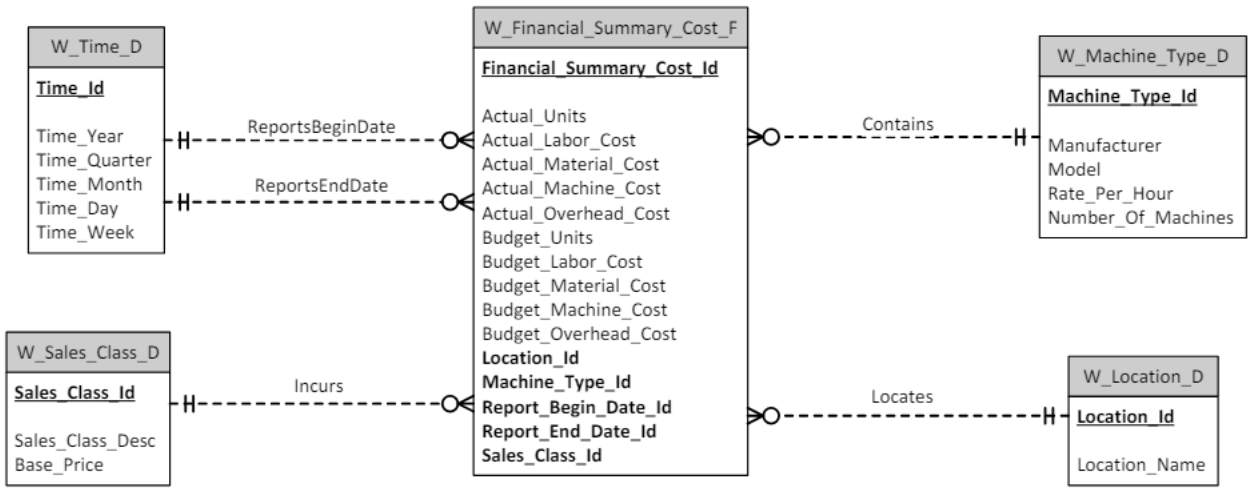
Hình 3.7. Sales Lead Schema

3.5.5. Invoice Schema



Hình 3.8. Invoice Schema

3.5.6. Cost financial summary schema



Hình 3.9. Cost financial summary schema

CHƯƠNG 4. NHỮNG YÊU CẦU ĐỐI VỚI CÁC TRUY VẤN PHÂN TÍCH

4.1. Nhu cầu Trí tuệ Kinh doanh

Mục đích chính của kho dữ liệu là hỗ trợ các truy vấn BI để theo dõi và so sánh doanh số và chi phí cho các phân nhánh chính theo các khoảng thời gian.

- **Doanh số:** Theo dõi theo địa điểm, loại hình bán hàng, nhân viên bán hàng và khách hàng.
- **Tỷ lệ thành công của khách hàng tiềm năng:** Theo dõi theo khách hàng, loại hình bán hàng và nhân viên bán hàng.
- **Chi phí:** Theo dõi theo địa điểm, loại hình bán hàng và loại máy móc. Chi phí nên được phân thành các thành phần trực tiếp (nhân công, vật liệu và máy móc) và gián tiếp (chi phí chung) để tập trung vào các điểm kém hiệu quả.
- **Lợi nhuận:** Kết hợp doanh số và chi phí, theo dõi theo khách hàng, loại hình bán hàng và địa điểm.
- **Doanh số, chi phí, tỷ lệ giữa các loại chi phí và lợi nhuận (số tiền, biên lợi nhuận và biên lợi nhuận gộp):** Theo dõi theo thời gian.

4.2. Yêu cầu thứ hai đối với bảng truy vấn và phân tích dữ liệu

Liên quan đến kiểm soát chất lượng cho hiệu suất theo hợp đồng và chất lượng sản xuất.

- Một công việc (job) là một hợp đồng với hai ngày quan trọng: ***ngày hứa hẹn*** trong đó toàn bộ số lượng công việc sẽ được giao và ***ngày giao hàng*** trong đó lô hàng đầu tiên của một công việc sẽ diễn ra.
  + Ngày hứa hẹn cung cấp ràng buộc về ngày giao hàng cuối cùng, trong khi ngày giao hàng cung cấp ràng buộc về ngày giao hàng đầu tiên.
  + Các truy vấn BI nên theo dõi độ lệch so với ngày giao hàng đầu tiên và cuối cùng theo địa điểm, loại hình bán hàng và khách hàng theo thời gian.
- **Trả hàng:** Được đo bằng sự khác biệt giữa số lượng đã giao và được xuất hóa đơn, là bằng chứng về chất lượng sản phẩm kém. Số lượng đã giao và được xuất hóa đơn giống nhau ngoại trừ khi hàng trả lại làm giảm số lượng được xuất hóa đơn. Các truy vấn BI nên theo dõi số lượng và số tiền trả hàng theo địa điểm, loại hình bán hàng và khách hàng theo thời gian.

4.3. Yêu cầu thứ ba đối với bảng truy vấn và phân tích dữ liệu

Liên quan đến hiệu suất phân tích tài chính và đối chiếu.

- Hiệu suất của các nhà phân tích tài chính trong việc dự báo và lập ngân sách cần được theo dõi.
- Dự báo được thực hiện theo loại hình bán hàng và địa điểm, trong khi lập ngân sách được thực hiện theo địa điểm, loại hình bán hàng và loại máy móc.

- Tổng hợp tài chính cho doanh số và chi phí liên quan đến việc đối chiếu.
  + Đối với doanh số, đối chiếu loại bỏ các giao dịch nội bộ liên quan đến doanh số giữa các địa điểm.
  + Đối với lập ngân sách, đối chiếu loại bỏ các chi phí trùng lặp liên quan đến máy móc thực hiện công việc chung cho nhiều công việc ở một số địa điểm.
  + Các truy vấn BI nên cho biết số tiền và nguồn của cả hai loại đối chiếu theo thời gian.

4.4. Các loại truy vấn

Mỗi khu vực này đều liên quan đến các truy vấn cơ bản tóm tắt các mối quan hệ trong các bảng kho dữ liệu cũng như các truy vấn phân tích sử dụng xếp hạng, so sánh theo cửa sổ, phân phối tích lũy và đóng góp của đơn vị kinh doanh cho toàn bộ công ty. Bài tập cung cấp chi tiết về các yêu cầu cho

4.5. Các Vấn đề về Mô hình Dữ liệu trong Kho dữ liệu Tập đoàn Thẻ CPI

4.5.1. Những lưu ý về Thiết kế Bảng

- Khi phân tích các yêu cầu cho từng vấn đề, bạn cần lưu ý đến một khía cạnh độc đáo của thiết kế bảng cho kho dữ liệu của nghiên cứu điển hình. Sơ đồ có các kết nối giữa các bảng dữ liệu chính như một biểu diễn các sự kiện liên quan trong quy trình kinh doanh cơ bản. Bạn nên nghiên cứu các bảng dữ liệu chính để xem các cột đại diện cho mối quan hệ giữa các công việc và tiểu công việc, tiểu công việc và lô hàng, và lô hàng và hóa đơn.
- Các mối quan hệ này không được duy trì dưới dạng khóa ngoại mặc dù có thể thiết lập các ràng buộc toàn vẹn tham chiếu.
- **Kết hợp các Bảng Dữ liệu Chính**

Nếu một vấn đề yêu cầu dữ liệu từ nhiều sự kiện kinh doanh, bạn nên suy nghĩ kỹ về việc kết hợp các bảng dữ liệu chính được liên kết. Ví dụ: một vấn đề có thể yêu cầu phân tích số lượng đặt hàng và số lượng được xuất hóa đơn. Bạn phải đảm bảo rằng các câu lệnh SELECT của bạn không có hoạt động tích chéo giữa các bảng dữ liệu chính. Bỏ qua điều kiện JOIN sẽ dẫn đến tích chéo, một hoạt động tiêu tốn tài nguyên thừa quá mức và tạo ra kết quả không chính xác. Nếu bạn cần kết hợp các bảng không liên quan trực tiếp, bạn cần sử dụng các bảng trung gian.

- **Cấu trúc Dữ liệu Kho dữ liệu**

Kho dữ liệu chứa dữ liệu tổng hợp, được tạo bởi phần mềm tạo dữ liệu. Dữ liệu được tạo là tổng hợp vì phần mềm sử dụng trình tạo số ngẫu nhiên Oracle để chọn các giá trị khóa ngoại và tạo các giá trị số trong các phạm vi được chỉ định. Phần mềm tạo dữ liệu cung cấp tính linh hoạt cho các tham số bao gồm khoảng thời gian, số lượng các chiều và tỷ lệ giữa các bảng dữ liệu chính (dẫn đến công việc, công việc đến tiểu công việc, tiểu công việc đến lô hàng và lô hàng đến hóa đơn), đồng thời duy trì tính nhất quán như mối quan hệ giữa số lượng công việc và số lượng tiểu công việc.

– **Phân bố Dữ liệu Ngẫu nhiên**

Mặc dù mối quan hệ giữa dữ liệu và chiều cùng với dữ liệu số thường được tạo ngẫu nhiên trong hầu hết các trường hợp, kho dữ liệu được điền thông tin vẫn có sự thiên lệch trong một số lĩnh vực. Trình tạo số ngẫu nhiên Oracle sử dụng phân phối đều, nghĩa là mọi giá trị trong một phạm vi đều có khả năng xảy ra như nhau. Hầu hết các giá trị số trong kho dữ liệu được điền thông tin đều được lấy từ phân phối đều mà không có sự thiên lệch. Trong một số trường hợp, phần mềm tạo dữ liệu đặc biệt tạo ra sự thiên lệch trên các giá trị khóa ngoại từ các bảng chiều. Ví dụ: một số loại hình bán hàng hoặc địa điểm có nhiều khả năng có dự báo kém hoặc vấn đề về kiểm soát chất lượng.

CHƯƠNG 5. PHÂN TÍCH DỮ LIỆU CHO CPT CARD

5.1. Xây dựng các truy vấn cơ sở

Đối với phần chính của nhiệm vụ này, bạn nên hiểu nhu cầu kinh doanh thông minh và liên hệ các nhu cầu đó với các bảng kho dữ liệu. Đối với nhu cầu thông tin nghiệp vụ, bạn nên đọc tài liệu liên quan cung cấp thông tin cơ bản về cách xây dựng truy vấn phân tích. Sau khi bạn hiểu rõ về cách biểu diễn cơ sở dữ liệu cho một vấn đề, bạn nên bắt đầu viết các câu lệnh SQL. Để giúp cấu trúc công thức truy vấn của bạn, bài tập chứa các truy vấn cơ sở không có hàm phân tích và các truy vấn có hàm phân tích. Trước tiên, bạn nên tạo các truy vấn cơ sở để phát triển các yêu cầu về biểu diễn cơ sở dữ liệu. Sau khi các truy vấn cơ sở của bạn thực hiện chính xác, bạn nên sửa lại chúng cho các hàm phân tích.

5.1.1. Base query 1

Location/Salesclass summary for job quantity and amount(revenue/costs)

Query (Area)	Result Columns	Comments
BQ1: Location/Sales class summary for job quantity and amount (revenue/costs)	Location id, location name, sales class id, sales class description, year of job contract date, month of job contract date, base price of sales class, sum of quantity ordered, sum of job amount	Job amount defined as quantity ordered times unit price; combine with time dimension table on contract date FK

Code SQL

location_saleClass_summary RUN SAVE QUERY (CLASSIC) SHARE SCHEDULE MORE

```sql
#create VIEW final-project-dw-423502.CardAPI.view_summary_salesClass_Location_for_jobQuantityAndAmnout AS
select
A.location_id
 ,B.location_name
 ,D.time_year AS contract_date_year
 ,D.time_month AS contract_date_month
 ,C.sales_class_id
 ,C.sales_class_desc
 ,C.base_price
 ,SUM(A.quantity_ordered) AS sum_quantity_ordered
 ,SUM(A.quantity_ordered *A.unit_price) AS sum_job_amount

 FROM
 final-project-dw-423502.CardAPI.w_Job_f A

 JOIN final-project-dw-423502.CardAPI.w_location_d B
 ON A. location_id = B.location_id

 JOIN final-project-dw-423502.CardAPI.w_Sales_Class_d C
 ON A.sales_class_id = C.sales_class_id

 JOIN final-project-dw-423502.CardAPI.w_Time_d D
 ON A.contract_date = D.time_id

 GROUP BY A.location_id,B.location_name, contract_date_year
 , contract_date_month ,C.sales_class_id ,C.sales_class_desc,C.base_price

 ORDER BY A.location_id,B.location_name, contract_date_year
 , contract_date_month,C.sales_class_id,C.sales_class_desc
 ,C.base_price
```

Result:

Query results

SAVE RESULTS | EXPLORE DATA

JOB INFORMATION | RESULTS | CHART | JSON | EXECUTION DETAILS | EXECUTION GRAPH

Row	location_id	location_name	contract_date_year	contract_date_month	sales_class_id	sales_class_desc	base_price	sum_quantity_ordered	sum_job_amount
1	1	New York	2012	1	1	Debit Smart	1.6	298500	468645.0
2	1	New York	2012	1	2	Credit Smart	1.6	981200	1461988.0
3	1	New York	2012	1	5	Prepaid NoSmart	0.8	571300	439901.0
4	1	New York	2012	1	6	Loyalty NoSmart	0.8	895800	662892.0
5	1	New York	2012	2	1	Debit Smart	1.6	1813100	2756736.0
6	1	New York	2012	2	2	Credit Smart	1.6	4599100	6834614.0
7	1	New York	2012	2	3	Debit NoSmart	0.8	2314100	1757434.0
8	1	New York	2012	2	4	Credit NoSmart	0.8	1964300	1532797.0
9	1	New York	2012	2	5	Prepaid NoSmart	0.8	772100	579075.0
10	1	New York	2012	2	6	Loyalty NoSmart	0.8	1775000	1321310.0
11	1	New York	2012	3	1	Debit Smart	1.6	5406800	8122172.0

Results per page: 50 | 1 – 50 of 3583

Job history | REFRESH

5.1.2. Base query 2

Location invoice revenue summary(revenue/costs)

BQ2: Location invoice revenue summary (revenue/costs)	Job id, location id, location name, job unit price, job order quantity, year of contract date, month of contract date, sum of invoice amount, sum of invoice quantity	Need to combine some fact tables; Be careful about cross product operations from missing a join condition among fact tables; Contract year/month used to match revenues and costs; Use a CTE or CREATE VIEW statement in addition to the SELECT statement.

Code SQL:

location_invoice_revenue_summary | RUN | SAVE QUERY (CLASSIC) | SHARE

```
#create VIEW final-project-dw-423502.CardAPI.view_summary_location_invoice_revenue AS
select
A.job_id,
A.location_id
 ,E.location_name
 ,A.unit_price
 ,A.quantity_ordered
 ,F.time_year AS contract_date_year
 ,F.time_month AS contract_date_month
 ,SUM(D.invoice_amount) AS sum_invoice_amount
 ,SUM(D.invoice_quantity) AS sum_invoice_quantity
FROM
 final-project-dw-423502.CardAPI.w_Job_f A
join final-project-dw-423502.CardAPI.w_Sub_Job_f B
ON A.job_id = B.job_id

JOIN final-project-dw-423502.CardAPI.w_Job_Shipment_f C
ON B.sub_job_id = C.sub_job_id

JOIN final-project-dw-423502.CardAPI.w_Invoice_Line_f D
ON C.invoice_id = D.invoice_id

JOIN final-project-dw-423502.CardAPI.w_location_d E
 ON A.location_id = E.location_id

JOIN final-project-dw-423502.CardAPI.w_Time_d F
ON A.contract_date = F.time_id

 GROUP BY A.job_id,A.location_id,E.location_name,A.unit_price
 ,A.quantity_ordered,F.time_year ,F.time_month
```

Result:

Query results

SAVE RESULTS EXPLORE DATA

JOB INFORMATION | RESULTS | CHART | JSON | EXECUTION DETAILS | EXECUTION GRAPH

Row	job_id	location_id	location_name	unit_price	quantity_ordered	contract_date_year	contract_date_month	sum_invoice_amount	sum_invoice_quantity
1	360096	8	Toronto	0.79	202700	2013	3	160133	202700
2	360097	2	Atlanta	1.6	63800	2015	2	102080	63800
3	360098	11	London	1.6	104500	2013	11	167200	104500
4	360099	5	Denver	1.57	341900	2013	7	483874	291034
5	360100	11	London	0.74	856000	2014	7	609316	823400
6	360101	3	Chicago	0.76	585600	2014	3	390868	503605
7	360102	9	Montreal	1.58	199500	2014	12	210140	133000
8	360103	10	Vancouver	0.74	833100	2014	7	462352	598592
9	360104	5	Denver	1.58	174400	2015	2	275552	174400
10	360105	11	London	1.54	534000	2012	3	792638	514700
11	360106	5	Denver	0.79	173100	2013	11	91166	115400
12	360107	1	New York	0.73	967000	2013	1	679484	906122

Results per page: 50 1 – 50 of 16759

Job history REFRESH

5.1.3. Base query 3

Location subjob cost summary (revenue/costs)

BQ3: Location subjob cost summary (revenue/costs)	Job id, location id, location name, year of contract date, month of contract date, sum of labor cost, sum of material cost, sum of machine cost, sum of overhead costs, sum of total costs, sum of quantity produced, unit cost	Need to combine some fact tables; machine costs computed as machine hours times rate per hour for machine; create a view; Contract year/month used to match revenues and costs; Unit cost defined as total costs / sum of quantity produced. Use a CTE or CREATE VIEW statement in addition to the SELECT statement.

Code SQL:

bq3_summary_location_subjob_cost RUN SAVE QUERY (CLASSIC) SHARE SCHEDULE MORE This query will proce

```
#create VIEW final-project-dw-423502.CardAPI.view_location_subjob_cost_summary AS
select
A.job_id,
C.location_id,C.location_name,
D.time_year AS contract_date_year
 ,D.time_month AS contract_date_month,
COUNT (A.job_id) as NumSubJob,
SUM(B.cost_labor) as sum_labor_cost,
SUM(B.cost_material)as sum_meterial_cost,
SUM(B.cost_overhead) AS sum_overhead_cost,
SUM(B.machine_hours * E.rate_per_hour) AS sum_machine_cost,
(SUM(B.cost_labor) + SUM(B.cost_material) + SUM(B.cost_overhead) + SUM(B.machine_hours * E.rate_per_hour)) AS total_cost,
SUM(B.quantity_produced) AS sum_quantity_produced,
ROUND ((SUM(B.cost_labor) + SUM(B.cost_material) + SUM(B.cost_overhead) + SUM(B.machine_hours * E.rate_per_hour)) / SUM(B.quantity_produced),2) AS unit_cost

FROM final-project-dw-423502.CardAPI.w_Job_f A

JOIN final-project-dw-423502.CardAPI.w_Sub_Job_f B ON A.job_id = B.job_id

JOIN final-project-dw-423502.CardAPI.w_location_d C ON A. location_id = C.location_id

JOIN final-project-dw-423502.CardAPI.w_Time_d D ON A.contract_date = D.time_id

JOIN final-project-dw-423502.CardAPI.w_machineType_d E ON B.machine_type_id = E.machine_type_id

GROUP BY A.job_id,C.location_id,C.location_name,
 D.time_year ,D.time_month

ORDER BY A.job_id,C.location_id,
C.location_name,D.time_year ,D.time_month
```

Result:

Query results

SAVE RESULTS | EXPLORE DATA

JOB INFORMATION | RESULTS | CHART | JSON | EXECUTION DETAILS | EXECUTION GRAPH

Row		location_id	location_name	contract_date_year	contract_date_month	NumSubJob	sum_labor_cost	sum_meterial_cost	sum_overhead_cost	sum_machine_cost	total_cost	sum_quantity_produ
1	360096	8	Toronto	2013	3	1	37150.86	3169.1	5170.63	18560	64050.59	202700
2	360097	2	Atlanta	2015	2	1	11841.28	6760.64	8604.44	13620	40826.36	63800
3	360098	11	London	2013	11	1	28758.4	13628.47	7338.41	17150	66875.28	104500
4	360099	5	Denver	2013	7	3	110022.4599999...	18214.39	15258.33999999...	71148	214643.19	341900
5	360100	11	London	2014	7	6	107637.44	41389.77	41242.82999999...	63020	253290.0399999...	856000
6	360101	3	Chicago	2014	3	3	86756.73999999...	25100.87	14785.96000000...	51310	177953.57	585600
7	360102	9	Montreal	2014	12	1	92041.32	15159.21	9691.95	9180	126072.48	199500
8	360103	10	Vancouver	2014	7	3	120690.75	25541.52	20236.14	80070	246538.4099999...	833100
9	360104	5	Denver	2015	2	1	82665.6	6199.92	14466.48	6870	110202.0	174400
10	360105	11	London	2012	3	3	165800.1	34716.89	37209.36	91170	328896.35	534000
11	360106	5	Denver	2013	11	1	38836.72	5902.58	3773.78	6184	54697.08	173100
12	360107	1	New York	2013	1	3	125216.6	56361.64	31442.58	69260	282280.82	967000

Results per page: 50 1 – 50 of 16912

Job history REFRESH

5.1.4. Base query 4

Returns by location and sales class (qualitycontrol)

BQ 4: Returns by location and sales class (quality control)	Location id, location name, sales class id, sales class description, year of invoice sent date, month of invoice sent date sum of quantity returned, sum of dollar amount of returns	Return quantity defined as quantity shipped minus quantity invoiced; condition for positive return quantity; use columns in the invoice fact table to calculate the unit price

Code SQL:

BQ 4 RUN SAVE QUERY (CLASSIC) SHARE SCHEDULE MORE

```
#create VIEW final-project-dw-423502.CardAPI.view_returns_by_sales_location_class AS
SELECT
  A.location_id
 ,B.location_name
 ,D.time_year AS invoice_sent_year
 ,D.time_month AS invoice_sent_month
 ,C.sales_class_id
 ,C.sales_class_desc
 ,SUM(A.quantity_shipped - A.invoice_quantity) AS sum_quantity_returned
 ,SUM(A.invoice_amount) AS sum_amount_return
FROM
 final-project-dw-423502.CardAPI.w_Invoice_Line_f A
JOIN
 final-project-dw-423502.CardAPI.w_location_d B ON A.location_id = B.location_id
JOIN
 final-project-dw-423502.CardAPI.w_Sales_Class_d C ON A.sales_class_id = C.sales_class_id
JOIN
 final-project-dw-423502.CardAPI.w_Time_d D ON A.invoice_sent_date = D.time_id
GROUP BY
  A.location_id
 ,B.location_name
 ,invoice_sent_year
 ,invoice_sent_month
 ,C.sales_class_id
 ,C.sales_class_desc
ORDER BY
  A.location_id
 ,invoice_sent_year
 ,invoice_sent_month
 ,C.sales_class_id;
```

Result:

Query results

SAVE RESULTS | EXPLORE DATA

JOB INFORMATION | RESULTS | CHART | JSON | EXECUTION DETAILS | EXECUTION GRAPH

Row	location_id	location_name	invoice_sent_year	invoice_sent_month	sales_class_id	sales_class_desc	sum_quantity_return	sum_amount_return
1	1	New York	2012	2	1	Debit Smart	0	93729
2	1	New York	2012	2	2	Credit Smart	0	1560716
3	1	New York	2012	2	4	Credit NoSmart	0	61383
4	1	New York	2012	2	5	Prepaid NoSmart	0	63063
5	1	New York	2012	2	6	Loyalty NoSmart	71582	559736
6	1	New York	2012	3	1	Debit Smart	0	435430
7	1	New York	2012	3	2	Credit Smart	40136	1582767
8	1	New York	2012	3	3	Debit NoSmart	35614	716158
9	1	New York	2012	3	4	Credit NoSmart	62981	998448
10	1	New York	2012	3	5	Prepaid NoSmart	35408	572061
11	1	New York	2012	3	6	Loyalty NoSmart	6256	226724
12	1	New York	2012	4	1	Debit Smart	152401	6834369

Results per page: 50 | 1 – 50 of 2788

Job history | REFRESH

5.1.5. Base query 5

Last shipment delays involving datepromised (qualitycontrol)

BQ5: Last shipment delays involving date promised (quality control)	Job id, location id, location name, sales class id, sales class description, time id of the date promised for the job, time id of the last shipment date, order quantity in the job, sum of shipped quantity after job promised date, difference in business days between last shipment date and promised date	Condition for last shipment date later than job promised date; use function provided for computing difference in business days; Use a CTE or CREATE VIEW statement in addition to the SELECT statement; See SELECT statements on pages 3 and 4

Code SQL:

bq5_Last_shipment_delays

RUN | SAVE QUERY (CLASSIC) | SHARE | SCHEDULE

```sql
#create VIEW final-project-dw-423502.CardAPI.view_shipment_delays_involving_date_promised AS
select
A.location_id,
D.location_name,
C.job_id,C.date_promised AS date_promised,
E.sales_class_id, E.sales_class_desc,
MAX(A.actual_ship_date) AS last_shipment_date,
SUM(A.actual_quantity) AS sum_delay_ship_qty,C.quantity_ordered,
DATE_DIFF (
  PARSE_DATE('%Y%m%d', CAST(MAX(A.actual_ship_date) AS STRING)),
  PARSE_DATE('%Y%m%d', CAST(C.date_promised AS STRING )),
DAY) AS days_diff_last_shipment_promised

FROM final-project-dw-423502.CardAPI.w_Job_Shipment_f A

JOIN final-project-dw-423502.CardAPI.w_Sub_Job_f B ON A.sub_job_id = B.sub_job_id

JOIN final-project-dw-423502.CardAPI.w_Job_f C ON B.job_id =C.job_id

JOIN final-project-dw-423502.CardAPI.w_location_d D ON C.location_id = D.location_id

JOIN final-project-dw-423502.CardAPI.w_Sales_Class_d E ON A.sales_class_id =E.sales_class_id

WHERE A.actual_ship_date=C.date_promised

GROUP BY A.location_id,D.location_name,C.job_id,C.date_promised,
E.sales_class_id, E.sales_class_desc,C.quantity_ordered

ORDER BY A.location_id,D.location_name,C.job_id,C.date_promised,
E.sales_class_id, E.sales_class_desc,C.quantity_ordered
```

Result:

Query results

SAVE RESULTS | EXPLORE DATA

JOB INFORMATION | RESULTS | CHART | JSON | EXECUTION DETAILS | EXECUTION GRAPH

Row	location_id	location_name	job_id	date_promised	sales_class_id	sales_class_desc	last_shipment_date	sum_delay_ship_qty	quantity_ordered	days_diff_last_shipp
1	1	New York	361704	20140116	6	Loyalty NoSmart	20140116	388500	1280000	0
2	1	New York	362157	20120412	4	Credit NoSmart	20120412	318800	784900	0
3	1	New York	363457	20120312	2	Credit Smart	20120312	211200	1035200	0
4	1	New York	364232	20150120	6	Loyalty NoSmart	20150120	91800	827700	0
5	1	New York	364310	20140611	2	Credit Smart	20140611	208200	592800	0
6	1	New York	364837	20130516	3	Debit NoSmart	20130516	52500	894500	0
7	1	New York	365615	20130710	3	Debit NoSmart	20130710	132000	803000	0
8	1	New York	366864	20130125	3	Debit NoSmart	20130125	398200	676700	0
9	1	New York	367026	20131017	5	Prepaid NoSmart	20131017	245900	329300	0
10	1	New York	367783	20120508	1	Debit Smart	20120508	68800	779500	0
11	1	New York	369469	20140206	3	Debit NoSmart	20140206	173900	673700	0

Results per page: 50 | 1 – 50 of 397

Job history | REFRESH

5.1.6. Base query 6

Firstshipment delays involving shipped by date (qualitycontrol)

BQ 6: First shipment delays involving shipped by date (quality control)	Job id, Location id, location name, sales class id, sales class description, time id of the shipped by date of the job, time id of the first shipment date, difference in business days between first shipment date and contractual shipped by date	Condition for first shipment date later than job shipped by date; use function provided for computing difference in business days; USE a CTE or CREATE VIEW statement in addition to the SELECT statement; See SELECT statements on pages 3 and 4

Code SQL:

bq6_first_shipment_ship_by RUN SAVE QUERY (CLASSIC) SHARE SCHEDULE

```sql
create VIEW final-project-dw-423502.CardAPI.view_shipment_delays_involving_date_shipped AS
select
A.location_id,
D.location_name,
C.job_id,C.date_ship_by  AS date_ship_by,
E.sales_class_id, E.sales_class_desc,
MIN(A.actual_ship_date) AS first_shipment_date,
SUM(A.actual_quantity) AS sum_delay_ship_qty,C.quantity_ordered,
DATE_DIFF (
  PARSE_DATE('%Y%m%d', CAST(MAX(A.actual_ship_date) AS STRING)),
  PARSE_DATE('%Y%m%d', CAST(C.date_ship_by AS STRING )),
DAY) AS days_diff_first_shipment_ship_by

FROM final-project-dw-423502.CardAPI.w_Job_Shipment_f A

JOIN final-project-dw-423502.CardAPI.w_Sub_Job_f B ON A.sub_job_id = B.sub_job_id

JOIN final-project-dw-423502.CardAPI.w_Job_f C ON B.job_id =C.job_id

JOIN final-project-dw-423502.CardAPI.w_location_d D ON C.location_id = D.location_id

JOIN final-project-dw-423502.CardAPI.w_Sales_Class_d E ON A.sales_class_id =E.sales_class_id

GROUP BY A.location_id,D.location_name,C.job_id,C.date_ship_by,
E.sales_class_id, E.sales_class_desc,C.quantity_ordered

ORDER BY A.location_id,D.location_name,C.job_id,C.date_ship_by,
E.sales_class_id, E.sales_class_desc,C.quantity_ordered
```

Result:

Query results SAVE RESULTS EXPLORE DATA

JOB INFORMATION | RESULTS | CHART | JSON | EXECUTION DETAILS | EXECUTION GRAPH

Row	location_id	location_name	job_id	date_ship_by	sales_class_id	sales_class_desc	first_shipment_date	sum_delay_ship_qty	quantity_ordered	days_diff_first_shipp
1	1	New York	360107	20130201	4	Credit NoSmart	20130115	967000	967000	-11
2	1	New York	360136	20120315	2	Credit Smart	20120305	362300	362300	-9
3	1	New York	360144	20120328	2	Credit Smart	20120301	1220900	1221300	-12
4	1	New York	360145	20120803	6	Loyalty NoSmart	20120718	796200	796200	-1
5	1	New York	360171	20121204	6	Loyalty NoSmart	20121106	1210100	1210100	-14
6	1	New York	360176	20140109	3	Debit NoSmart	20131219	216100	216100	-8
7	1	New York	360177	20140217	4	Credit NoSmart	20140117	279200	279200	-4
8	1	New York	360179	20121017	5	Prepaid NoSmart	20120913	1220400	1220700	-16
9	1	New York	360191	20130410	2	Credit Smart	20130321	639700	639700	-19
10	1	New York	360194	20140221	3	Debit NoSmart	20140203	858500	858500	-9
11	1	New York	360211	20121227	1	Debit Smart	20121204	591400	591700	-15
12	1	New York	360225	20150114	5	Prepaid NoSmart	20150106	501200	501200	-5

Results per page: 50 1 – 50 of 16912

Job history REFRESH

5.2. Xây dựng các truy vấn phân tích

5.2.1. Truy vấn phân tích liên quan đến xu hướng doanh thu của khách hàng

5.2.1.1 Analytic query 1

Cumulative quantity for locations

Analytic Query	Analytic Functions	Other Columns	Notes
AQ1: Cumulative quantity for locations	Cumulative sum of amount ordered; Restart cumulative sum by location name and year; Use contract month as the ordering criteria; cumulative sum involves current row and all preceding rows	Location name, contract year, contract month, sum of job amount ordered	Extends location/sales class summary (BQ1)

Code SQL:

aq1_cumulative_quantity_for...ons RUN SAVE QUERY (CLASSIC) SHARE

```
1  SELECT
2    location_name,
3    contract_date_year,
4    contract_date_month,
5    sum_job_amount,
6    SUM(sum_job_amount)
7  OVER (PARTITION BY location_name, contract_date_year
8  ORDER BY contract_date_month
9  ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW) AS cumulative_sum_job_amount
10
11
12   FROM `final-project-dw-423502.CardAPI.view_summary_salesClass_Location_for_jobQuantityAndAmnout`
```

Result:

Query results

JOB INFORMATION | RESULTS | CHART | JSON | EXECUTION DETAILS | EXECUTION GRAPH

Row	location_name	contract_date_year	contract_date_month	sum_job_amount	cumulative_sum_job
1	Los Angeles	2015	1	5687319.0	5687319.0
2	Los Angeles	2015	1	2063306.0	7750625.0
3	Los Angeles	2015	1	10176302.0	17926927.0
4	Los Angeles	2015	1	7313858.0	25240785.0
5	Los Angeles	2015	1	2762695.0	28003480.0
6	Los Angeles	2015	1	2108122.0	30111602.0
7	Los Angeles	2015	2	2301758.0	32413360.0
8	Los Angeles	2015	2	6966654.0	39380014.0
9	Los Angeles	2015	2	2412168.0	41792182.0
10	Los Angeles	2015	2	2089249.0	43881431.0
11	Los Angeles	2015	2	4672181.0	48553612.0
12	Los Angeles	2015	2	3460377.0	52013989.0

5.2.1.2 Analytic query 2

Moving average of average amount ordered for locations

AQ2: Moving average of average amount ordered for locations	Moving average of average amount ordered; Restart moving average by location name; Use combination of contract year and month as the ordering criteria; moving average of current row and 11 preceding rows	Location name, contract year, contract month, average of job amount ordered	Extends location/sales class summary (BQ1)

Code SQL:

```sql
SELECT
  location_name,
  contract_date_year,
  contract_date_month,
  ROUND(AVG(sum_job_amount)
  OVER (PARTITION BY location_name
  ORDER BY contract_date_month, contract_date_year
  ROWS BETWEEN 11 PRECEDING AND CURRENT ROW),2) AS
  moving_avg_avg_job_amount

  FROM `final-project-dw-423502.CardAPI.view_summary_salesClass_Location_for_jobQuantityAndAmnout`
```

Result:

Query results

SAVE RESULTS | EXPLORE DATA

JOB INFORMATION | RESULTS | CHART | JSON | EXECUTION DETAILS | EXECUTION GRAPH

Row	location_name	contract_date_year	contract_date_month	moving_avg_avg_job
1	Vancouver	2012	1	1187768.0
2	Vancouver	2012	1	783958.5
3	Vancouver	2012	1	788751.0
4	Vancouver	2013	1	1507017.0
5	Vancouver	2013	1	1796466.4
6	Vancouver	2013	1	2097400.5
7	Vancouver	2013	1	2889596.14
8	Vancouver	2013	1	3636947.13
9	Vancouver	2013	1	3682976.89
10	Vancouver	2014	1	3608492.6
11	Vancouver	2014	1	3647354.91
12	Vancouver	2014	1	3516722.83

Results per page: 50 | 1 – 50 of 3583

Job history | REFRESH

5.2.2. Truy vấn phân tích liên quan đến tóm tắt doanh thu và chi phí

5.2.2.1 Analytic query 3

Rank locations by descending sum of annual profit

AQ3: Rank locations by descending sum of annual profit	Rank in descending order of annual sum of profit; Restart ranks for each year	Location name, contract year, contract month, sum of profit	Use CTEs or views for BQ2 and BQ3 in the FROM clause; Sum of profit calculated as sum of invoice amount minus sum of total costs (labor, material, machine, and overhead)

Code SQL:

```
SELECT
   A.location_name,
   A.contract_date_year,
   A.contract_date_month,
   SUM(A.sum_invoice_amount - B.total_cost) as sum_profit,
  RANK() OVER (PARTITION BY A.location_name ORDER BY SUM(A.sum_invoice_amount - B.total_cost)DESC) AS rank_loc_by_desc_sum_annual_profit
  FROM `final-project-dw-423502.CardAPI.view_summary_location_invoice_revenue`  A
  JOIN final-project-dw-423502.CardAPI.view_location_subjob_cost_summary B
  ON A.job_id = B.job_id
  GROUP BY A.location_name, A.contract_date_year, A.contract_date_month
```

Result:

JOB INFORMATION | RESULTS | CHART | JSON | EXECUTION DETAILS | EXECUTION GRAPH

Row	location_name	contract_date_year	contract_date_month	sum_profit	rank_loc_by_desc_su
1	New York	2013	9	618398395.1299...	1
2	New York	2014	8	584263512.03	2
3	New York	2014	2	582603301.2399...	3
4	New York	2014	12	555569115.2300...	4
5	New York	2013	5	555545205.42	5
6	New York	2012	9	543788163.05	6
7	New York	2014	11	540196979.4100...	7
8	New York	2014	4	536400695.0400...	8
9	New York	2013	3	516416957.0999...	9
10	New York	2014	6	494153294.04	10
11	New York	2013	12	492422714.2099...	11
12	New York	2014	9	483679750.36	12

5.2.2.2 Analytic query 4

Rank locations by descending annual profit margin

AQ4: Rank locations by descending annual	Rank in descending order of annual profit margin;	Location name, contract year, contract month,	Use CTEs or views for BQ2 and BQ3 in

profit margin	Restart ranks for each year	profit margin	the FROM clause; Profit margin calculated as annual sum of profit divided by annual sum of invoice amount; See AQ3 for calculation of sum of profit

Code SQL:

```sql
SELECT
   A.location_name,
   A.contract_date_year,
   A.contract_date_month,
   SUM ((A.sum_invoice_amount - B.total_cost) / A.sum_invoice_amount) as profit_margin,

  RANK() OVER (PARTITION BY A.contract_date_year ORDER BY (SUM(A.sum_invoice_amount - B.total_cost) / SUM(A.sum_invoice_amount))  DESC) AS
rank_loc_by_desc_annual_profit_margin
  FROM `final-project-dw-423502.CardAPI.view_summary_location_invoice_revenue`  A
  JOIN final-project-dw-423502.CardAPI.view_location_subjob_cost_summary B
  ON A.job_id = B.job_id
  GROUP BY A.location_name, A.contract_date_year, A.contract_date_month
  ORDER BY A.contract_date_year, profit_margin DESC;
```

Result:

JOB INFORMATION | RESULTS | CHART | JSON | EXECUTION DETAILS | EXECUTION GRAPH

Row	location_name	contract_date_year	contract_date_month	profit_margin	rank_loc_by_desc_ap
1	London	2012	8	47.19335483982...	35
2	Chicago	2012	6	47.05616917397...	55
3	Atlanta	2012	7	44.12878427662...	64
4	Los Angeles	2012	8	43.32323749358...	17
5	Seattle	2012	7	40.22834907199...	57
6	Vancouver	2012	7	39.15333089968...	82
7	Vancouver	2012	4	38.62342140925...	107
8	Toronto	2012	6	38.38980837710...	12
9	Seattle	2012	9	38.37362875985...	19
10	Vancouver	2012	8	38.36940806672...	21
11	Birmingham	2012	7	38.35768294295...	16
12	Atlanta	2012	11	38.25004145323...	54

5.2.2.3 Analytic query 5

Percent rank of job profit margins for locations

AQ5: Percent rank of job profit margins for locations	Percent rank of profit margin for jobs; Use all jobs so no restarting of percent ranks	Job id, location name, contract year, contract month, profit margin	Use CTES or views for BQ2 and BQ3 in the FROM clause; See AQ4 for calculation of profit margin

Code SQL:

aq5_percent_rank_profit_margin | RUN | SAVE QUERY (CLASSIC) | SHARE | SCHEDULE | MORE

```
#create VIEW final-project-dw-423502.CardAPI.view_percent_rank_profit_margin AS


SELECT
  A.location_name,
  A.contract_date_year,
  A.contract_date_month,
  A.job_id,
  SUM ((A.sum_invoice_amount - B.total_cost) / A.sum_invoice_amount) as profit_margin,
  PERCENT_RANK () OVER(ORDER BY ROUND (SUM ((A.sum_invoice_amount - B.total_cost) / A.sum_invoice_amount)),2) AS percent_rank_profit_margin

 FROM `final-project-dw-423502.CardAPI.view_summary_location_invoice_revenue` A
 JOIN final-project-dw-423502.CardAPI.view_location_subjob_cost_summary B
  ON A.job_id = B.job_id
  GROUP BY A.location_name,
  A.contract_date_year,
  A.contract_date_month,
  A.job_id

```

Result:

JOB INFORMATION | RESULTS | CHART | JSON | EXECUTION DETAILS | EXECUTION GRAPH

Row	location_name	contract_date_year	contract_date_month	job_id	profit_margin	percent_rank_profit_
1	Montreal	2012	2	373840	-11.6819515397...	0.0
2	Toronto	2012	2	373190	-9.41564538611...	7.170514842965...
3	Dallas	2012	2	360607	-8.04692456043...	0.000143410296...
4	New York	2012	2	373780	-6.22004041771...	0.000215115445...
5	Los Angeles	2012	2	368930	-5.58547136907...	0.000215115445...
6	Denver	2012	2	360160	-4.01134856874...	0.000358525742...
7	Atlanta	2012	2	362277	-3.63683759982...	0.000358525742...
8	Dallas	2012	2	366061	-4.36576149166...	0.000358525742...
9	Atlanta	2012	2	365835	-4.03677130136...	0.000358525742...
10	Toronto	2012	2	375193	-2.72721718998...	0.000645346335...
11	Vancouver	2012	2	362189	-3.33022370423...	0.000645346335...
12	Dallas	2012	2	363869	-2.86054290934...	0.000645346335...

Results per page: 5

5.2.2.4 Analytic query 6

Top performers of percent rank of job profit margins for locations

AQ6: Top performers of percent rank of job profit margins for locations	Percent rank of profit margin for jobs; Use all jobs so no restarting of percent ranks	Job id, location name, contract year, contract month, profit margin	Refinement of AQ5 to show only top 5% of job profit margins; Use AQ5 in the FROM clause or make a CTE for AQ5 and use the new CTE in the FROM clause

Code SQL:

```
SELECT *
 FROM `final-project-dw-423502.CardAPI.view_percent_rank_profit_margin`
 WHERE percent_rank_profit_margin > 0.95
```

Result:

Query results

JOB INFORMATION | RESULTS | CHART | JSON | EXECUTION DETAILS | EXECUTION GRAPH

There is no data to display.

5.2.3. Truy vấn phân tích liên quan đến trả về

5.2.3.1 Analytic query 7

Rank sales class by return quantities for each year

AQ7: Rank sales class by return quantities for each year	Rank in descending order of sum of return quantity; Restart ranks for each year of the date sent year	Sales class description, year of date sent, sum of return quantity	Extends return summary (BQ4)

Code SQL:

aq7_rank_sales_class _by_re...ear RUN SAVE QUERY (CLASSIC) SHARE SCHEDULE MORE

```
1  SELECT sales_class_desc, invoice_sent_year, SUM(sum_quantity_returned) as total_sum_quantity_returned
2  , RANK () OVER (PARTITION BY sales_class_desc, invoice_sent_year ORDER BY  SUM(sum_quantity_returned) DESC ) AS rank
3   FROM `final-project-dw-423502.CardAPI.view_returns_by_sales_location_class`
4  GROUP BY sales_class_desc, invoice_sent_year
5
```

Result:

JOB INFORMATION | RESULTS | CHART | JSON | EXECUTION DETAILS | EXECUTION GRAPH

Row	sales_class_desc	invoice_sent_year	total_sum_quantity	rank
1	Debit NoSmart	2012	12300274	1
2	Loyalty NoSmart	2014	16552342	1
3	Loyalty NoSmart	2015	4432822	1
4	Debit NoSmart	2015	3568609	1
5	Prepaid NoSmart	2015	3594160	1
6	Credit NoSmart	2012	11025107	1
7	Debit NoSmart	2013	14732361	1
8	Loyalty NoSmart	2013	16085464	1
9	Prepaid NoSmart	2012	11188279	1
10	Debit Smart	2014	16311612	1
11	Debit Smart	2015	4091654	1
12	Credit NoSmart	2015	3232417	1

5.2.3.2 Analytic query 8

Ratio to report of return quantities for sales classes by year;

Remember, RATIO_TO_REPORT not supported in PostgreSQL but simplework around

AQ8: Ratio to report of return quantities for sales classes by year; Remember, RATIO_TO_REPORT not supported in PostgreSQL but simple work around	Ratio to report of sum of return quantity; Restart ratios for each date sent year	Sales class description, year of date sent, sum of return quantity	Extends return summary (BQ4); order by year and return quantity

Code SQL:

aq8_ratio_to_report RUN SAVE QUERY (CLASSIC) SHARE SCHEDULE MORE

```
1  SELECT sales_class_desc, invoice_sent_year, SUM(sum_quantity_returned) as total_sum_quantity_returned
2  ,ROUND (SUM(sum_quantity_returned) / SUM(SUM(sum_quantity_returned)) OVER (PARTITION BY sales_class_desc), 3) as ratio_to_report
3   FROM `final-project-dw-423502.CardAPI.view_returns_by_sales_location_class`
4  GROUP BY sales_class_desc, invoice_sent_year
5  ORDER BY invoice_sent_year, total_sum_quantity_returned DESC
```

Result:

Query results SAVE RESULTS

JOB INFORMATION | RESULTS | CHART | JSON | EXECUTION DETAILS | EXECUTION GRAPH

Row	sales_class_desc	invoice_sent_year	total_sum_quantity_r	ratio_to_report
1	Loyalty NoSmart	2012	12308067	0.249
2	Debit NoSmart	2012	12300274	0.267
3	Debit Smart	2012	11782599	0.247
4	Credit Smart	2012	11620687	0.252
5	Prepaid NoSmart	2012	11188279	0.235
6	Credit NoSmart	2012	11025107	0.236
7	Credit NoSmart	2013	16183634	0.346
8	Loyalty NoSmart	2013	16085464	0.326
9	Prepaid NoSmart	2013	15554601	0.327
10	Debit Smart	2013	15550271	0.326
11	Debit NoSmart	2013	14732361	0.32
12	Credit Smart	2013	14014097	0.304

Results per page: 50 1 – 24 of 24

Job history

5.2.4. Truy vấn phân tích liên quan đến sự chậm trễ trong hợp đồng

5.2.4.1 Analytic query 9

Rank locations by sum of business days delayed for the job shipped by date (first shipment)

AQ9: Rank locations by sum of business days delayed for the job shipped by date (first shipment)	Rank in descending order of sum of business days delayed; Use both ranking functions; Restart rankings on year of shipped by date	Location name, year of date shipped by, sum of difference in business days	Use view for shipped date delays (BQ6); view should only contain delayed jobs

Code SQL:

aq9_rank_locations _by_sum_...ate

RUN | SAVE QUERY (CLASSIC) | SHARE | SCHEDULE

```
1  SELECT  location_name,
2  B.time_year as year_of_date_ship_by
3  , SUM(A.days_diff_first_shipment_ship_by) as sum
4  , RANK () OVER (PARTITION BY A.location_name ORDER BY SUM(A.days_diff_first_shipment_ship_by) DESC) AS rank
5  FROM `final-project-dw-423502.CardAPI.view_shipment_delays_involving_date_shipped` A
6  JOIN final-project-dw-423502.CardAPI.w_Time_d B
7  ON A.date_ship_by =B.time_id
8  GROUP BY location_name, year_of_date_ship_by
```

Result:

JOB INFORMATION | RESULTS | CHART | JSON | EXECUTION DETAILS | EXECUTION GRAPH

Row	location_name	year_of_date_ship_by	sum	rank
1	Toronto	2015	-970	1
2	Toronto	2012	-3368	2
3	Toronto	2013	-4223	3
4	Toronto	2014	-4615	4
5	Los Angeles	2015	-655	1
6	Los Angeles	2012	-3455	2
7	Los Angeles	2014	-3993	3
8	Los Angeles	2013	-4256	4
9	Montreal	2015	-622	1
10	Montreal	2012	-3131	2
11	Montreal	2014	-3993	3
12	Montreal	2013	-4418	4

Results per page: 50 1 – 48 of 48

Job history

5.2.4.2 Analytic query 10

Rank locations by delay rate for jobs delayed on the last shipment date

AQ10: Rank locations by delay rate for jobs delayed on the last shipment date	Rank in descending order of delay rate; Restart rankings on year of date promised	Location name, year of date promised, count of delayed jobs, sum of difference in business days	Use view for last promised date delays (BQ5); view should only contain delayed jobs

Code SQL:

aq10_rank_locations_by_dela...ate RUN SAVE QUERY (CLASSIC) SHARE SCHEDULE MORE

```sql
SELECT
  A.location_name,
  B.time_year as year_of_date_promised,
  COUNT (A.job_id) as amount_delayed_job, SUM(A.days_diff_last_shipment_promised) as sum,
  SUM(quantity_ordered - sum_delay_ship_qty) / SUM(quantity_ordered) ,
  RANK () OVER (PARTITION BY A.location_name ORDER BY SUM(quantity_ordered - sum_delay_ship_qty) / SUM(quantity_ordered) desc) as rank

 FROM `final-project-dw-423502.CardAPI.view_shipment_delays_involving_date_promised` A
 JOIN final-project-dw-423502.CardAPI.w_Time_d B
ON A.date_promised =B.time_id
GROUP BY A.location_name, B.time_year

```

Result:

Query results SAVE RESULTS

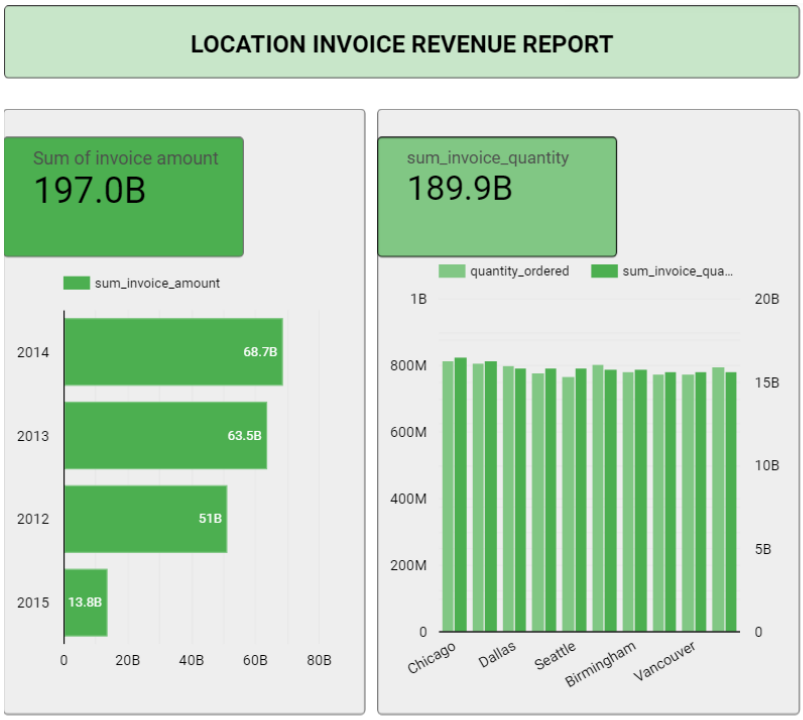
JOB INFORMATION | RESULTS | CHART | JSON | EXECUTION DETAILS | EXECUTION GRAPH

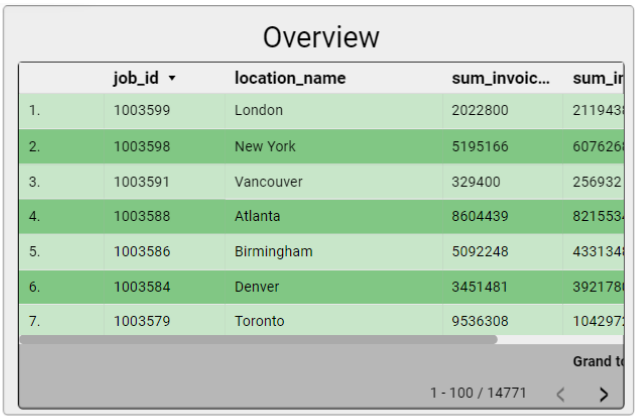
Row	location_name	year_of_date_promis	amount_delayed_job	sum	f0_	rank
1	Los Angeles	2015	4	0	0.875271216911...	1
2	Los Angeles	2013	16	0	0.829172094040...	2
3	Los Angeles	2014	20	0	0.815954037991...	3
4	Los Angeles	2012	9	0	0.747766935150...	4
5	Seattle	2013	25	0	0.796503291313...	1
6	Seattle	2015	8	0	0.786253709330...	2
7	Seattle	2012	17	0	0.773318855546...	3
8	Seattle	2014	20	0	0.728740315012...	4
9	Montreal	2012	12	0	0.852319600753...	1
10	Montreal	2015	8	0	0.833841995778...	2
11	Montreal	2013	16	0	0.773193387663...	3
12	Montreal	2014	12	0	0.744120911608...	4

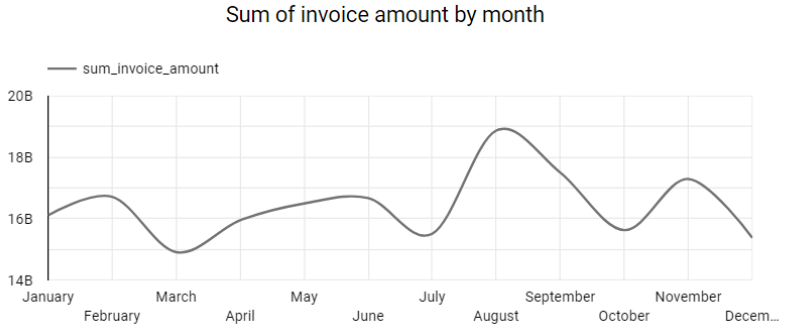
Results per page: 50 1 – 46 of 46

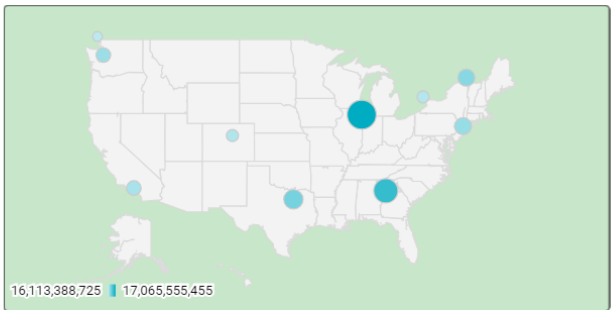
CHƯƠNG 6. TRỰC QUAN HÓA DỮ LIỆU CHO CPI CARD

[Trực quan hóa các dữ liệu của CPI CARD trên gg studio]









Location subjob cost summary

Sum of overhead cost

541.8M

Sum of total cost

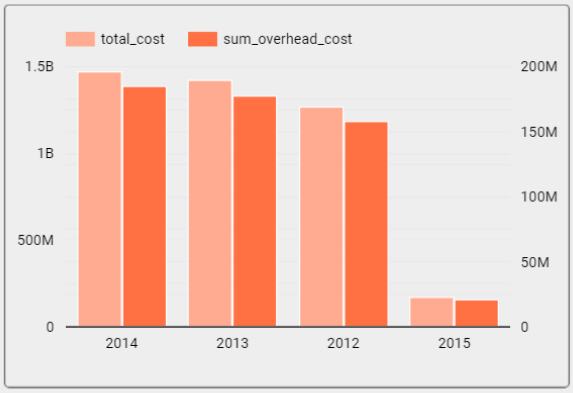
4.3B

	location_name	sum_labor_cost	sum_quantity_produced
1.	Vancouver	175.3M	864.9M
2.	Toronto	180.9M	901.5M
3.	Seattle	181.6M	879.7M
4.	New York	188.4M	930.3M
5.	Montreal	182M	916.5M
6.	Los Angeles	180.6M	923M
7.	London	177.7M	871.6M
8.	Denver	174.6M	869.5M

1 - 12 / 12

location_na...	sum_ove...
Type to search	
New York	46.4M
Dallas	46.3M
London	45.8M

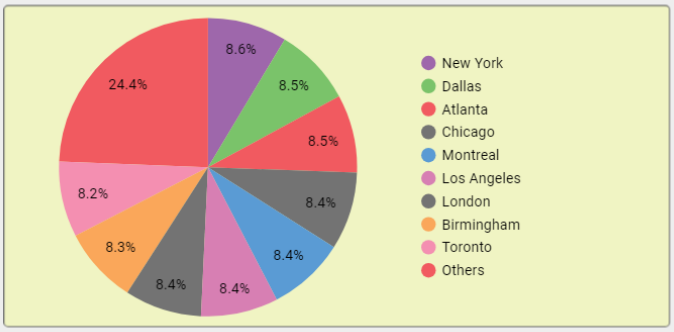
contract_da...	sum_ove...
Type to search	
2014	184.8M
2013	177.4M



	contract_date_year	sum_labor_cost
1.	2015	89.3M
2.	2014	734.7M
3.	2013	713M
4.	2012	633M

1 - 4 / 4

Percentage of machine cost



Sum of material cost