Date Warehouse Development of Car Repair Shop System

Group 3

Irina Becknell, Joe Chan, Thomas Jensen, Brandon Russell

University of Maryland University College

Author Note

This project was prepared for DBST665, taught by Professor Kuchibhotla

Data Warehouse Development of Car Repair Shop System

Abstract

In this project a data warehouse is developed for a car repair shop. The project begins with development of a project plan and a business case for the data warehouse. This will be followed by a design of a data warehouse schema, which consists of one FACT table and five DIMENSION tables. Farther, the requirements are gathered and analyzed. Then, the database is designed and built. The research concludes with summary of findings and lessons learned. This research is meant to investigate the data warehouse technology.

**Project Plan**

Table 1 presents the work breakdown structure (WBS) of the project.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Phase** | **Steps** | **Week** | **Task** | **Group member** |
| 1 | 1 | 1 | Group formed | All |
| 2 | 1 | Select the topic | All |
| 3 | 1 | Roles assignment | All |
| 4 | 2 | Develop project plan (WBS) for the data warehouse project | Irina Becknell |
| 5 | 2 | Inform professor of the project topic and submit group work schedule to professor | Irina Becknell |
| 6 | 3 to 12 | Inform professor of the project progress and clarify any questions with professor | Irina Becknell |
| 7 | 4 | Writing of the Abstract of the project report | Irina Becknell |
| 8 | 4 | Writing of the Introduction of the project report | Irina Becknell |
| 2 | 9 | 4 | Data Model/DDL design (DB schema consisting of one FACT table and five DIMENSION tables). Record and include DDL script design for Appendix | Brandon Russell |
| 10 | 4 | Requirements analysis (develop description of the data sources, analyze the requirements documented in the introduction.) | Irina Becknell |
| 11 | 4 | ETLs/DMLs (identification of the data sources, data map, and the methodology used to load the data. Identify, describe and record the DML or SQL\*Loader scripts for Appendix ) | Joe Chan |
| 12 | 6 | Data Model/DDL development and implementation. Record steps for the body of the report | Brandon Russell |
| 13 | 7 | Writing SQL queries. Record the scripts for the Appendix | Brandon Russell |
| 14 | 8 | Report analysis (development of all the report scenarios (at least 4) that demonstrate how the requirements identified in the introduction were met. Identify and record the SQL scripts in the Appendix). | Thomas Jensen |
| 3 | 15 | 10 | Writing of Lesson learned section (record any problems and issues) | Irina Becknell |
| 16 | 10 | Writing of Conclusion of the project report (one page discussion of the project, explain how has this report solved the business needs identified in the Introduction.) | Irina Becknell |
| 17 | 11 | Review of the project report, make necessary changes. Submission of the project | All |

Table 1. Project plan (WBS)

The following roles were assigned to the team members (Table 2):

|  |  |
| --- | --- |
| **Team Member** | **Role** |
| Becknell, Irina | Project Lead |
| Chan, Joe | Business Analyst |
| Jensen, Thomas | Database Administrator (DBA) |
| Russell, Brandon | Database Developer |

Table 2. Team members and roles

**Introduction**

**Requirements Analysis**

Requirements analysis is the first stage in the systems engineering process and software development process: it encompasses those tasks that go into determining the needs or conditions to meet for a new or altered product, taking account of the possibly conflicting requirements of the various stakeholders, such as beneficiaries or users. Requirements analysis is critical to the success of a development project (Software Development Process – activities and steps, 2010).

**Introduction**

* **Purpose of the system** The purpose of the developed data warehouse system is to provide tools for reporting and data analysis of a hypothetical car repair shop. The car repair shop data warehouse system that is to be developed in this project is considered a core component of business intelligence.

The database will enable a user to enter customer’s personal data (first and last names, address, phone number), vehicle’s data (make, model, year, color, body type, number of axles, fuel type, number of cylinders, weight), parts (name, description, category, brand, weight, unit of issue), maintenance actions (name, description, category, number of hours spent, team size), time (time name, day, month, year, quarter. The reporting data will be used to track business metrics, such as cost, time spent on repairs, quarterly sales, etc.

* **Scope of the system**

The project scope is determined as follows:

* Data Model/DDL design – database schema consisting of five dimension tables (CUSTOMERS, MAINTENACE\_ACTIONS, TIME\_TABLE, VEHICLES, PARTS) and one fact table (WORK\_FACT). Data base schema is to be developed and DDL files will be created to build the database
* ETL/DML development

Data source

The source of the data for the CUSTOMERS table is the data that was directly taken from the customers at the time of the car repair estimate. The data source of the PARTS table is the data that comes from the manufacturer of the automobile parts. The data source of the VEHICLES table comes from the car manufacturers and the customer if the car is not already recorded in the database. The data source of the MAINTENANCE\_ACTIONS is supplied by the employees and occasionally company president. The TIME table data source is provided by the database administrator who must create data entries representing moments in time. The WORK\_FACT table data is received from the store employee before and after the car repair is completed. For this project, a sample of random data is created for each table and it is stored the data in CSV files (Customers.csv, Maintenance\_Actions.csv, Parts.csv, Time\_Table.csv, Vehicels.csv, Work.csv).

The dimension tables are as follows:

1. Customer

The data comes from a relational database system. This dimension is intended to represent individuals that have previously paid for services or parts from the auto repair shop or their partners. For instance, if the customer has purchases parts or services from an associated shop, the auto shop will have access to these records.

The identifying customer information instance includes:

* Customer Id
* Customer First Name
* Customer Last Name
* Customer Address
* Customer City
* Customer Country
* Customer State
* Customer Zip Code
* Customer Phone Number

1. Maintenance Actions

The data comes from a relational database system. This dimension represents the maintenance actions that the auto repair shop offers based on parts and expertise. Mechanics log the type of maintenance and number of hours to complete the job into the system using an API, along with a description of the work done and any notes.

Each maintenance action type instance includes:

* Maintenance Action ID
* Maintenance Name
* Maintenance Action Description (Oil Change, battery replacement, wheel alignment, coolant flush)
* Maintenance Action Category (Engine, brakes, transmission, electrical)
* Maintenance number of hours required
* Maintenance Team Size Required

1. Parts

The data comes from a relational database system. This dimension keeps track the parts available in the warehouse along the supplier (brand) that the part is manufactured by. The availability of parts needs to be known for mechanics to be able to perform certain maintenance actions.

Each Parts instance includes:

* Part ID
* Part Name
* Part Description
* Part Category
* Part Brand
* Part Weight
* Parts Union of Issue available in the warehouse

1. Vehicles

The data comes from a relational database system. This dimension represents the attributes of all vehicles currently in circulation. Vehicle dates range from 1983 to current day. The system needs to be updated regularly to include the attributes of newly released vehicles in order to order corresponding parts and further training for mechanics.

Each Vehicle instance includes:

* Vehicle ID
* Vehicle Make
* Vehicle Model
* Vehicle Year
* Vehicle Body Type
* Number of Axles
* Fuel Type
* Number of Cylinders
* Vehicle Weight

1. Time

The data comes from a relational database system. This dimension represents the attributes of time of repairs that were performed by the car repair shop.

* Time ID
* Time Day
* Time Month Year
* Time Month
* Time Month Number
* Time Year
* Time Day of Week
* Time Quarter
* Time Julian (i.e. Julian time metric)

Fact Table

The fact table is Work Fact. It consists of the foreign keys (FK) that correspond to the primary keys (PK) in the corresponding dimension table.

* FK Cust\_ID (correspond to PK Cust\_ID in CUSTOMERS)
* FK Parts\_ID (correspond to PK Parts\_ID in PARTS)
* FK MainAct\_ID (correspond to PK MainAct\_ID in MAINTENANCE\_ACTIONS)
* FK Veh\_ID (correspond to PK Veh\_ID in VEHICLES)
* FK Time\_ID (correspond to PK Time\_ID in TIME)
* Work odometer reading
* Work Cost
* Work Number of Action Hours
* SQL Queries creation – SQL queries is to be created for the reporting purposes
* **Current system**

There is no current system that the hypothetical car repair shop uses.

* **Technical specifications**

**Diagram Tool**

ER Assistant will be used to design ERD of the database. ERD will communicate all entities of the database, its attributes and keys; it will also identify the relationships between entities in the database.

**Database**

Object-relational database management systemOracle 11g (via web service Amazon Work Space (AWS)) will be used to develop and implement the database. Oracle 11g will be also used to create the tables, SQL queries and other required database objects for the project via SQL DDL statements.

**Hardware and Software**

Environment that will be used to implement the project is UMUC’s AWS Intel(R) Xeon(R) with operating system Microsoft Windows Server 2012R2.

**DDL and DML**

SQL will be used for the DDL and DML. SQL will be used to translate the information captured in the ER diagram (created in ER Assistant) into the database table creation scripts via DDL scripts. SQL Loader scripts will be used to pre-populate and load the tables of the database. Python script will be used to generate data for the Work\_Fact table.

Schedule/Milestones

* Stage 1 - Project planning phase – 01/29/2018-02/23/2018
* Stage 2 - Design phase (development, building, implementation, testing and reporting) – 02/26/2018-03/23/2018
* Stage 3 – Project closure phase – 03/26/2018-04/15/2018

**Data Model/DDL Development**

The database schema consists of five dimension tables (CUSTOMERS, MAINTENACE\_ACTIONS, TIME\_TABLE, VEHICLES, PARTS) and one fact table (WORK\_FACT).

The relationships between tables are as follows: the fact table has a one to many (1:M) relationship with each dimension table, i.e. there are 5 relationships defined.

Each dimension table has a primary key (PK) which uniquely identifies each row in the corresponding table. For example, each customer is uniquely identified in the CUSTOMER table by Cust\_ID.

WORK\_FACT table consists of all the PKs from the dimension tables (Cust\_ID, Parts\_ID, MaintACT\_ID, Veh\_ID, Time\_ID) and some additional metrics (cost, etc.) These PKs in the WORK\_FACT table are foreign keys (FK) that each correspond to the PK of its table. For example, Cust\_ID in the WORK\_FACT table is a FK corresponding to PK Cust\_ID in the CUSTOMERS table.

The database schema is presented in Figure 1. The DDL script that is used to build the database is attached in the Appendix A and attached to the project (DBST665\_Group\_Project\_DDLs.sql).

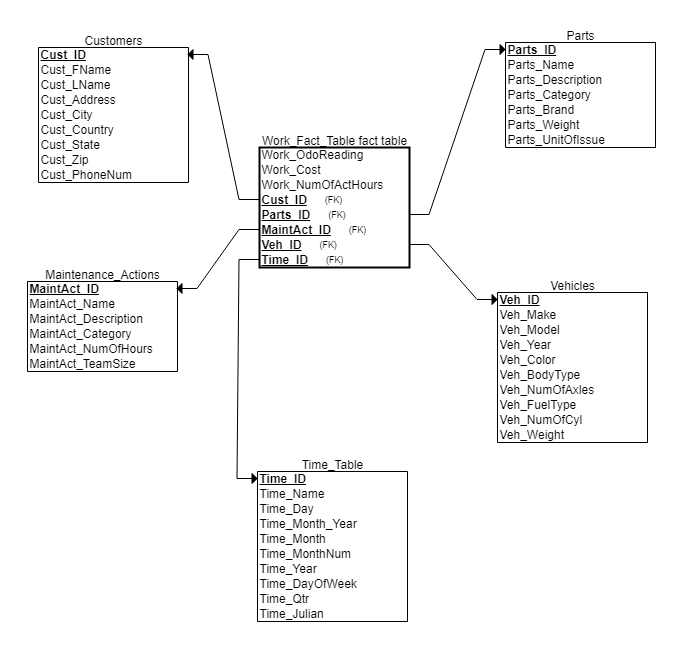


Figure 1. Database Schema

**ETL/DML Development**

SQL Loader scripts are used to pre-populate and load the database. There are six SQL loader scripts used to load each dimension table and the fact table. The sqlldr command is used in SQL\*Plus to upload the data from a csv file using commands instructed by a control file. A bad file was created to capture errors in the data upload.

Loading the Customer dimension table is presented in Figure 2.

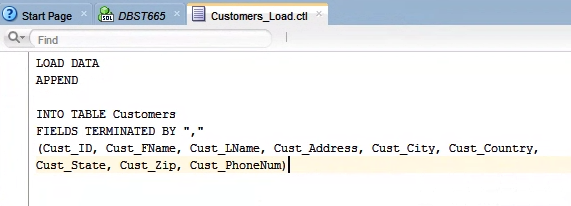


Figure 2. CUSTOMER table population - Step 1

The Customers\_Load.ctl file loaded the data into the Customers dimension table using the APPEND function. The data was loaded from the Customers.csv file into columns of the Customers table labeled Cust\_ID, Cust\_FName, Cust\_LName, Cust\_Address, Cust\_City, Cust\_Country, Cust\_State, Cust\_Zip, and Cust\_PhoneNum.

The SQLLDR command identified the userid and password at the database, the control file, the csv data file, and the bad file for the customers load procedure (Figure 3).



Figure 3. CUSTOMER table population - Step 2

Loading the Vehicles dimension table is shown in Figure 4.

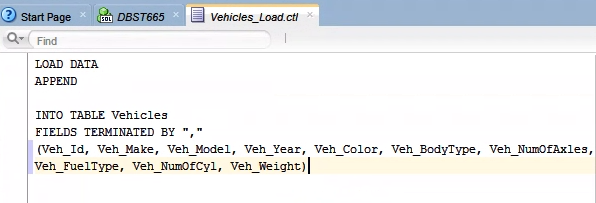


Figure 4. VEHICLES table population - Step 1

The Vehicles\_Load.ctl file loaded the data into the Vehicles dimension table using the APPEND function. The data was loaded from the Vehicles.csv file into columns of the Vehicles table labeled Veh\_Id, Veh\_Make, Veh\_Model, Veh\_Year, Veh\_Color, Veh\_BodyType, Veh\_NumOfAxles, Veh\_FuelType, Veh\_NumOfCyl, Veh\_Weight.

The SQLLDR command identified the userid and password at the database, the control file, the csv data file, and the bad file for the vehicles load procedure (Figure 5).

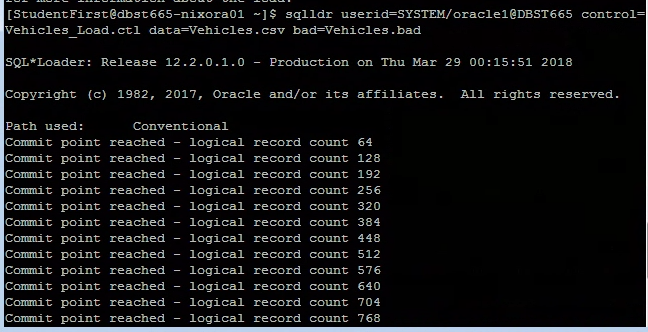


Figure 5. VEHICLES table population - Step 2

Loading the Parts dimension table is presented in Figure 6.

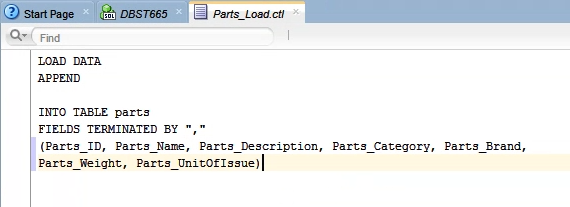


Figure 6. PARTS table population - Step 1

The Parts\_Load.ctl file loaded the data into the parts dimension table using the APPEND function. The data was loaded from the Parts.csv file into columns of the parts table labeled Parts\_ID, Parts\_Name, Parts\_Description, Parts\_Category, Parts\_Brand, Parts\_Weight, and Parts\_UnitOfIssue.

The SQLLDR command identified the userid and password at the database, the control file, the csv data file, and the bad file for the parts load procedure (Figure 7).



Figure 7. PARTS table population - Step 2

Next, The Maintenance Actions dimension table is loaded (Figure 8)

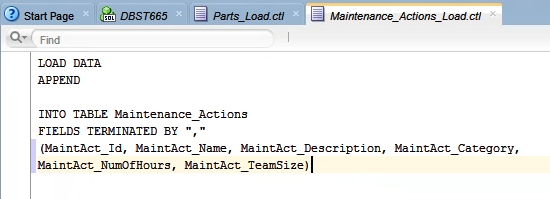


Figure 8. MAINTENANCE ACTIONS table population - Step 1

The Maintenance\_Actions\_Load.ctl file loaded the data into the Maintenance\_Actions dimension table using the APPEND function. The data was loaded from the Maintenance\_Actions.csv file into columns of the Maintenance\_Action table labeled MaintAct\_Id, MainAct\_Name, MaintAct\_Description, MaintAct\_Category, MaintAct\_NumOfHours, and MaintAct\_TeamSize.

The SQLLDR command identified the userid and password at the database, the control file, the csv data file, and the bad file for the Maintenance Actions load procedure (Figure 9).



Figure 9. MAINTENANCE ACTIONS table population - Step 2

Loading the Time Table dimension table is presented in Figure 10.

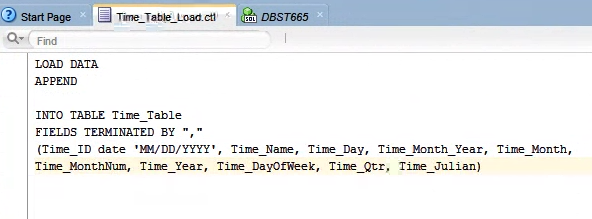


Figure 10. TIME table population - Step 1

The Time\_Table\_Load.ctl file loaded the data into the Time\_Table dimension table using the APPEND function. The data was loaded from the Time\_Table.csv file into columns of the Time\_Table table labeled Time\_ID, Time\_Name, Time\_Day, Time\_Month\_Year, Time\_Month, Time\_MonthNum, Time\_Year, Time\_DayOfWeek, Time\_Qtr, and Time\_Julian.

The SQLLDR command identified the userid and password at the database, the control file, the csv data file, and the bad file for the time table load procedure (Figure 11).



Figure 11. TIME table population - Step 2

Finally, we load the Work Fact table (Figure 12)

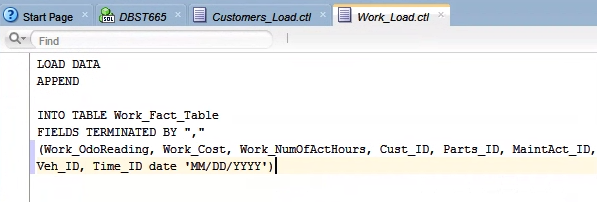


Figure 12. WORK FACT table population - Step 1

The Work\_Load.ctl file loaded the data into the Work\_Fact\_Table fact table using the APPEND function. The data was loaded from the Work.csv file into columns of the Work\_Fact\_Table table labeled Work\_OdoReading, Work\_Cost, Work\_NumOfActHours, Cust\_ID, Parts\_ID, MaintAct\_ID, Veh\_ID, and Time\_ID.

The SQLLDR command identified the userid and password at the database, the control file, the csv data file, and the bad file for the work fact table load procedure (Figure 13).



Figure 13. WORK FACT table population - Step 2.

**SQL Queries**

Now the database is ready for the reporting. The following SQL queries are developed for the reporting purposes (the scripts for all SQL queries presented below are attached in Appendix B and attached to the project DBST665\_SQLQueries.sql):

1. Suppose we want to check to see how many parts with a weight of more than 20 are not being used. We run the command below to find that amount (Figure 14).

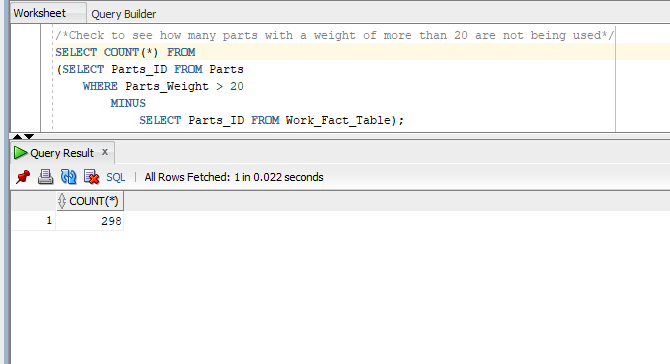


Figure 14. Amount of unused parts with a weight of more than 20

1. Let’s assume the car repair shop wants to send out promotional coupons. Hence we would like to get the list of repeated customers, so we can send them the coupon as they most likely will use it. The result of the query is presented in Figure 15.

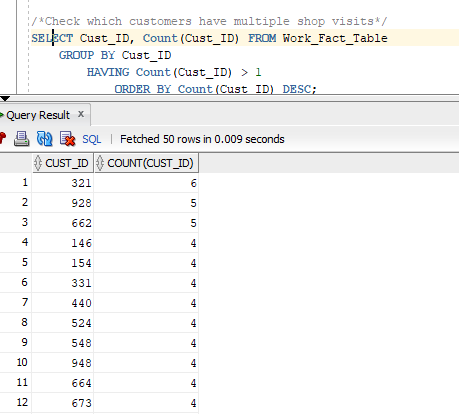


Figure 15. List of repeated customers

1. In order to perform reconciliation check, we run the command below (Figure 16).

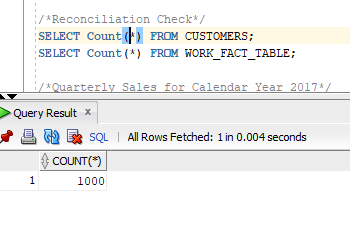


Figure 16. reconciliation check

1. One other common requested report is the Quarterly Sales. Let’s review the Quarterly Sales for Calendar Year 2017 (Figure 17).

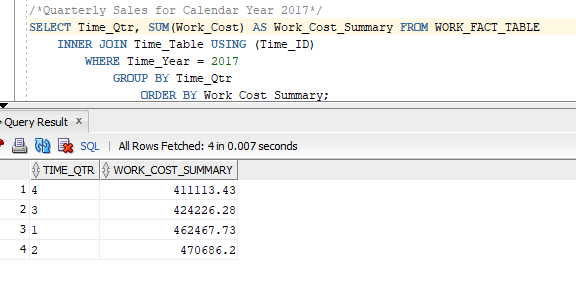
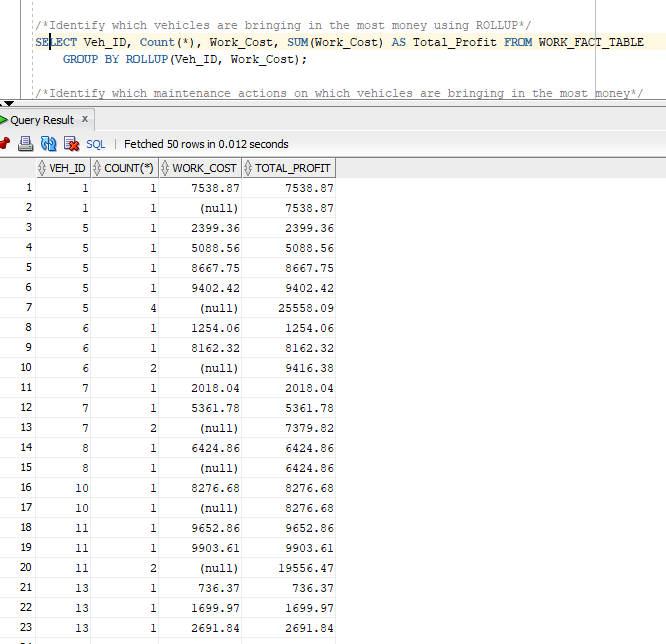


Figure 17. Quarterly Sales for 2017

1. Management is also always interested what brings most business and produces most profit for the company. In our case, we can find that out by identifying which vehicles are bringing in the most profit by using ROLLUP function (Figure 18)

Figure 18. vehicles are bringing in the most profit

1. We also could check which tasks and on which vehicles generate the most profit for the car repair shop (Figure 19).

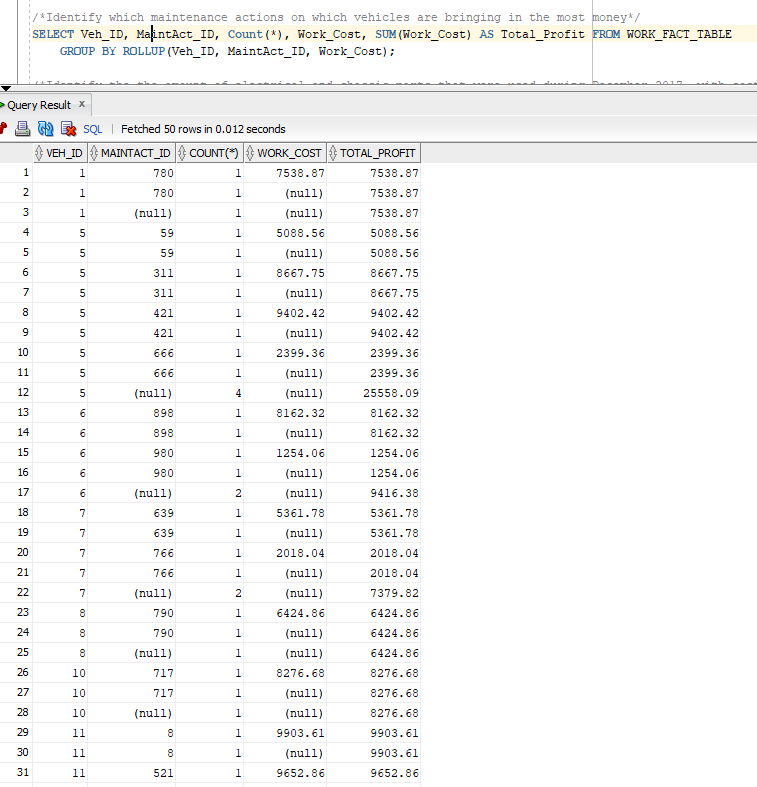
****

Figure 19. Tasks and vehicles generating the most profit for the car repair shop

1. The query below shows the amount and cost of electrical and chassis parts that were during December 2017 (Figure 20)

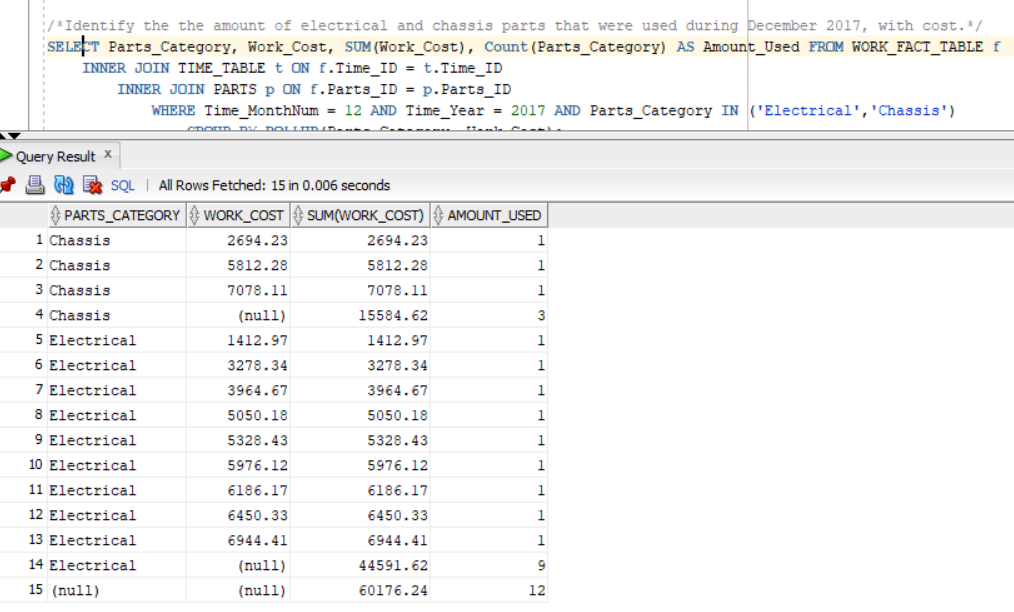


Figure 20. The amount and cost of electrical and chassis parts that were during December 2017

1. The query below returns the amount of maintenance actions taken along with the number of individual hours worked and a sum of all hours for each maintenance actions and overall (Figure 21).

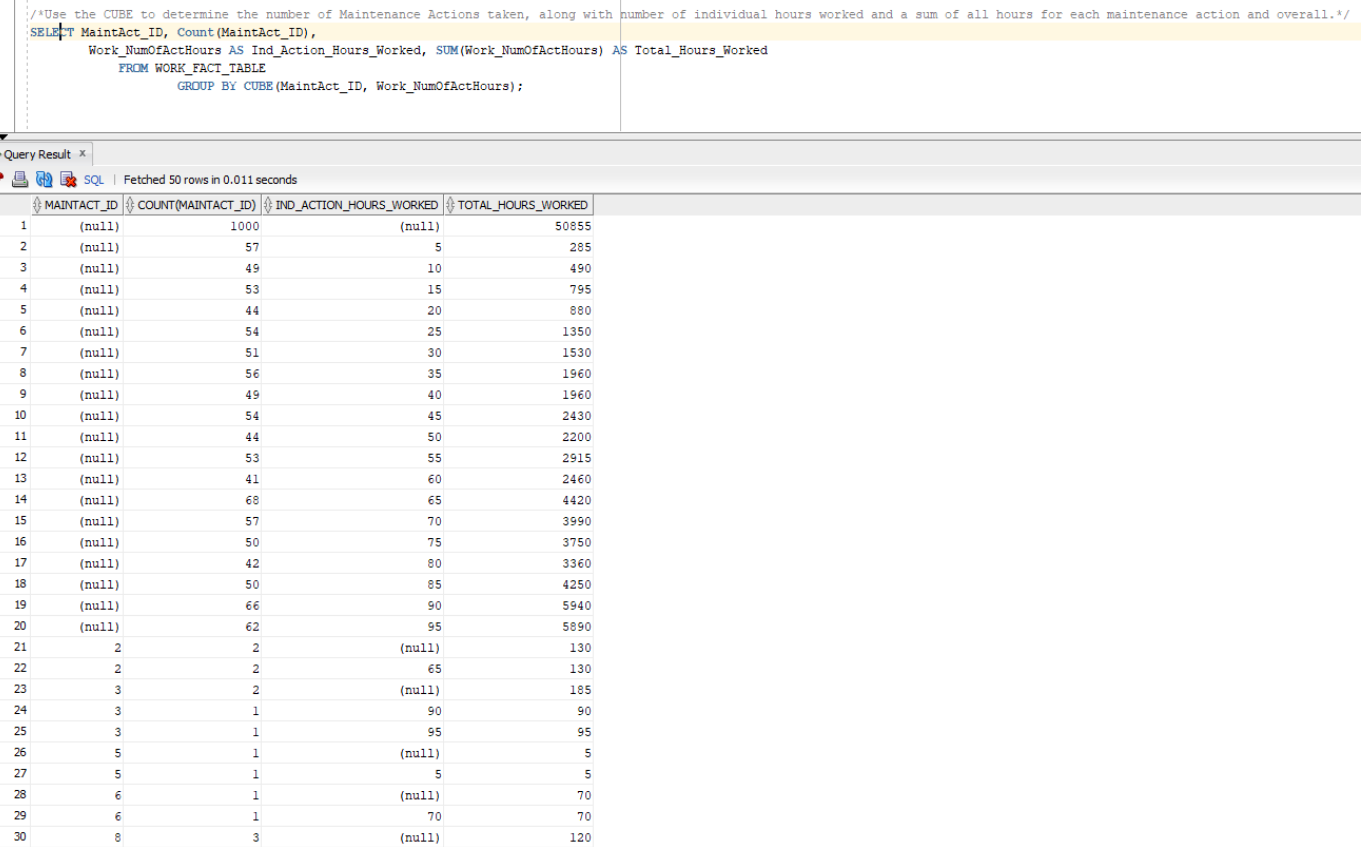


Figure 21. Amount of maintenance actions taken along with the number of individual hours worked and a sum of all hours for each maintenance actions and overall

**Report Analysis**

The requirement for the developed data warehouse system is to provide tools to the management of hypothetical car repair shop that will be used for reporting and data analysis. Based on that data the management would make educated business intelligence decisions in order to increase productivity and profitability of the company and decrease waste.  The system is designed to allow users to enter information pertaining to customers, vehicles, parts, and maintenance actions during specific time periods. This data is then used to report on business metrics including number of parts used, cost of parts and repairs, time spent on maintenance, and monthly, quarterly, and yearly sales.  This information provides business value to the car repair shop by keeping track of parts associated with maintenance actions, identifying customers for targeted marketing campaigns, and tracking sales.

For our reports we determined we would need nine separate queries, as demonstrated in the previous section. The first query demonstrated our ability to query our system for parts that are not being used. This is a useful metric for determining what parts we do or do not need to keep on hand for cost purposes. The second query showcases our ability to see which customers frequented our shops, providing us the ability to provide focus marketing, such as coupons, towards these customers. The third and fourth queries simply provided reconciliation checks. The fifth query showed quarterly sales data for the calendar year 2017. This is a useful metric in determining which quarters held better sales. The sixth query highlights which vehicles frequented the shop, and which provided the shop the most profit. This could be used in analysis to determine which vehicles make the most money. The seventh query determined which maintenance actions on different vehicles provided the most profit, providing similar metrics as the sixth query. The eighth query shows amounts used for specific category of parts and profits made. This is useful data for maintaining stocks, in general, and keeping stock of profitable items. The final query used the CUBE function to provide an analysis of how many times different maintenance actions were worked, along with how many hours were worked on each one and overall. This is a useful metric if we need to determine what training our employees need the most. More specifically, the four scenarios below verify that the business requirements of the system are being met:

* Scenario 1:

The system will be used to determine which parts need to be kept in stock because they are required for popular maintenance actions.  The test query for scenario 1 verifies that system can track parts that are associated with specific maintenance actions. In the test scenario the threshold that is a popular maintenance action is considered to be more than 5 (as see in Figure 14).  This threshold would be higher in an operational system, however when using test data, this is considered high.

* Scenario 2:

The system will be used to determine monthly and quarterly sales for a specific year including the total sales for an entire year (Figure 17).  The query used for scenario 2 verifies that the system can report cost results for a time period. This report provides information on the total earnings of the auto repair shop during each month and quarter.  That information can be used to identify times of year when sales (work\_cost) are high, as well as times of the year when sales are low. The car repair shop management could use the marketing strategies to promote more sales during low earning points in the year.  The CUBE function provides a summary of the total earnings for the year.

* Scenario 3:

The system was used to determine customers to receive a promotion that have repeatedly bought a specific maintenance action (Figure 15).  This report can be used to encourage returning customers to continue getting their cars serviced at the car repair shop. This scenario verifies that maintenance actions can be tracked and used to identify customer spending habits.  The use of targeted marketing can be used to increase returning customers, and as a result increase profits.

* Scenario 4:

The system was used to determine which maintenance actions are the least profitable in order to send out coupons to customers to promote the sales of these maintenance actions and increase profits.  This report could be used by the managers in charge of marketing to encourage customers to come into the car repair shop and take advantage of discounted maintenance actions that require parts taking up space in the shop.

The SQL queries for the described above scenarios are included in Appendix C.

**Lessons Learned**

There were a few valuable lessons that the group has learnt during the project.

First, we learned an importance of determining the grain, identifying the metrics in the fact and dimension tables. During the creation of the data model we decided to model a car repair shop maintenance tracking system. While there are several statistically viable attributes, we determined that the most important metrics to track, and hence, place into the fact table are the odometer reading of the vehicle at the time of entry into the system, the cost of overall work, and finally, the number of actual hours spent to complete the job. The odometer reading would not work well in the vehicles table, since this would require an update to the reading on every new job, thus not allowing for an historical record of the mileage of the vehicle between jobs. Though individual cost could be entered into the dimension tables, this would again force us to update these tables, versus tracking trends in the historical fact table data. The number of actual hours in the fact table contrasts with the estimated hours for a job listed in the maintenance actions table. This is a viable metric for tracking shop efficiency.

Also, we learnt that the importance of the way the data is loaded to the tables cannot be overlooked. In our project, we previously tracked all the records in comma-delimited csv files. Therefore, the decision was made that the best method for loading our data into the database would be using the SQL loader and control files. The data sources in the form of CSV files were loaded into the dimension tables using SQL Loader scripts. This method was utilized because it did not require the creation of DML commands to add data one row at a time, which would have been time-consuming and less efficient. The CSV files were pre-populated with data. A lesson learned was that the data in the CSV files must be saved in the correct format, such as text, number, date, etc. The dimension table would expect data to match the correct datatype that was specified for the columns when the dimension table was created. Otherwise, an error occurs due to the mismatched datatypes. In addition to that, control files were created for each CSV file and its corresponding dimension table. The CTL file consisted of SQL commands loading the data into the corresponding table using the APPEND command. A lesson learned for the CTL files is that the column fields must be written exactly the same as the column fields in the CSV table, otherwise an error occurs because the program cannot match the columns for correct loading. In most cases, the error occurs because of a spelling mistake. Another lesson learned is that the CTL file must re-format the data from the CSV file into the correct format, such as the date datatype, during the loading of the data into the columns. If the CSV columns are not re-formatted into the correct datatype, such as date datatype for the Time\_Table table, then an error occurs, and none of the data is loaded.

Another lesson that was learnt is the importance of indexes. While creating the DDLs, we determined that there is a need for indexes on the foreign key constraints within the fact table so we can increase query speeds.

And finally, we learnt the importance of different functions when creating the SQL queries. While creating the queries we learned it is required to utilize the SQL ROLLUP and CUBE functions for effective statistical analysis of many of the car repair shop metrics.

**Conclusion**

Data warehouses are centralized data storage systems that allow the organization to integrate data from multiple applications and sources into one location, which provides an environment that is designed for decision support, analytics reporting, and data mining. When the data is isolated and optimized, it can be managed without impacting primary business processes.

The benefits of data warehousing are obvious: warehousing solves the ongoing problem of analyzing separate data and converting it into actionable information that can be used. Warehousing also allows for processing of large amounts of complex data in an efficient way. With a successful implementation of a data warehouse system, it is possible to access the benefits such as increased efficiency, profitability and overall success through ETL and data warehousing. In addition to that, warehouse system offers improved and timely access to information **with enhanced quality and consistency, increased query and system performance, which leads to higher return on the investments, as the** use of data warehousing also enables the business to generate a higher amount of revenue due to cost savings (The Benefits of Data Warehousing and ETL, 2016).

It is clear that when data warehousing is implemented and designed properly it can provide a variety of advantages to the business.

The presented data warehouse design would be extremely beneficial for the car repair company as it would bring numerous advantages such as improved data management, reduced speeds in critical decision making and therefore improved business operation.

**Appendixes**

1. **DDL Script to build the database(creating tables)**

DROP TABLE Work\_Fact\_Table;

DROP TABLE Parts;

DROP TABLE Customers;

DROP TABLE Maintenance\_Actions;

DROP TABLE Time\_Table;

DROP TABLE Vehicles;

CREATE TABLE Parts (

Parts\_ID NUMBER NOT NULL,

Parts\_Name VARCHAR(50) NOT NULL,

Parts\_Description VARCHAR(100) NOT NULL,

Parts\_Category VARCHAR(50) NOT NULL,

Parts\_Brand VARCHAR(50) NOT NULL,

Parts\_Weight NUMBER NOT NULL,

Parts\_UnitOfIssue NUMBER NOT NULL,

CONSTRAINT pk\_parts PRIMARY KEY (Parts\_ID)

);

CREATE TABLE Customers (

Cust\_ID NUMBER NOT NULL,

Cust\_FName VARCHAR(50) NOT NULL,

Cust\_LName VARCHAR(50) NOT NULL,

Cust\_Address VARCHAR(100) NOT NULL,

Cust\_City VARCHAR(50) NOT NULL,

Cust\_Country VARCHAR(50) NOT NULL,

Cust\_State VARCHAR(50) NOT NULL,

Cust\_Zip VARCHAR(50) NOT NULL,

Cust\_PhoneNum VARCHAR(50) NOT NULL,

CONSTRAINT pk\_customers PRIMARY KEY (Cust\_ID)

);

CREATE TABLE Maintenance\_Actions (

MaintAct\_ID NUMBER NOT NULL,

MaintAct\_Name VARCHAR(50) NOT NULL,

MaintAct\_Description VARCHAR(100) NOT NULL,

MaintAct\_Category VARCHAR(100) NOT NULL,

MaintAct\_NumOfHours NUMBER NOT NULL,

MaintAct\_TeamSize NUMBER NOT NULL,

CONSTRAINT pk\_maintenance\_actions PRIMARY KEY (MaintAct\_ID)

);

CREATE TABLE Time\_Table (

Time\_ID DATE NOT NULL,

Time\_Name VARCHAR(50)NOT NULL,

Time\_Day NUMBER NOT NULL,

Time\_Month\_Year VARCHAR(50)NOT NULL,

Time\_Month VARCHAR(50) NOT NULL,

Time\_MonthNum NUMBER NOT NULL,

Time\_Year NUMBER(4,0) NOT NULL,

Time\_DayOfWeek VARCHAR(50) NOT NULL,

Time\_Qtr VARCHAR(50) NOT NULL,

Time\_Julian NUMBER NOT NULL,

CONSTRAINT pk\_time PRIMARY KEY (Time\_ID)

);

CREATE TABLE Vehicles (

Veh\_ID NUMBER NOT NULL,

Veh\_Make VARCHAR(50) NOT NULL,

Veh\_Model VARCHAR(50) NOT NULL,

Veh\_Year NUMBER(4,0) NOT NULL,

Veh\_Color VARCHAR(50) NOT NULL,

Veh\_BodyType VARCHAR(50) NOT NULL,

Veh\_NumOfAxles NUMBER NOT NULL,

Veh\_FuelType VARCHAR(50) NOT NULL,

Veh\_NumOfCyl NUMBER NOT NULL,

Veh\_Weight NUMBER NOT NULL,

CONSTRAINT pk\_vehicles PRIMARY KEY (Veh\_ID)

);

CREATE TABLE Work\_Fact\_Table (

Work\_OdoReading NUMBER NOT NULL,

Work\_Cost NUMBER(\*, 2) NOT NULL,

Work\_NumOfActHours NUMBER NOT NULL,

Cust\_ID NUMBER NOT NULL,

Parts\_ID NUMBER NOT NULL,

MaintAct\_ID NUMBER NOT NULL,

Veh\_ID NUMBER NOT NULL,

Time\_ID DATE NOT NULL,

CONSTRAINT pk\_work\_fact\_table PRIMARY KEY (Cust\_ID,Parts\_ID,MaintAct\_ID,Veh\_ID,Time\_ID),

CONSTRAINT fk\_customers FOREIGN KEY (Cust\_ID)

REFERENCES Customers ON DELETE CASCADE,

CONSTRAINT fk\_maintenance\_actions FOREIGN KEY (MaintAct\_ID)

REFERENCES Maintenance\_Actions ON DELETE CASCADE,

CONSTRAINT fk\_time\_table FOREIGN KEY (Time\_ID)

REFERENCES Time\_Table ON DELETE CASCADE,

CONSTRAINT fk\_vehicles FOREIGN KEY (Veh\_ID)

REFERENCES Vehicles ON DELETE CASCADE,

CONSTRAINT fk\_parts FOREIGN KEY (Parts\_ID)

REFERENCES Parts ON DELETE CASCADE

);

CREATE INDEX fk\_customers on Work\_Fact\_Table(Cust\_ID);

CREATE INDEX fk\_maintenance\_actions on Work\_Fact\_Table(MaintAct\_ID);

CREATE INDEX fk\_time\_table on Work\_Fact\_Table(Time\_ID);

CREATE INDEX fk\_vehicles on Work\_Fact\_Table(Veh\_ID);

CREATE INDEX fk\_parts on Work\_Fact\_Table(Parts\_ID);

1. **SQL Queries**

/\*Check to see how many parts with a weight of more than 20 are not being used\*/

SELECT COUNT(\*) FROM

(SELECT Parts\_ID FROM Parts

WHERE Parts\_Weight > 20

MINUS

SELECT Parts\_ID FROM Work\_Fact\_Table);

/\*Check which customers have multiple shop visits\*/

SELECT Cust\_ID, Count(Cust\_ID) FROM Work\_Fact\_Table

GROUP BY Cust\_ID

HAVING Count(Cust\_ID) > 1

ORDER BY Count(Cust\_ID) DESC;

/\*Reconciliation Check\*/

SELECT Count(\*) FROM CUSTOMERS;

SELECT Count(\*) FROM WORK\_FACT\_TABLE;

/\*Quarterly Sales for Calendar Year 2017\*/

SELECT Time\_Qtr, SUM(Work\_Cost) AS Work\_Cost\_Summary FROM WORK\_FACT\_TABLE

INNER JOIN Time\_Table USING (Time\_ID)

WHERE Time\_Year = 2017

GROUP BY Time\_Qtr

ORDER BY Work\_Cost\_Summary;

/\*Identify which vehicles are bringing in the most money using ROLLUP\*/

SELECT Veh\_ID, Count(\*), Work\_Cost, SUM(Work\_Cost) AS Total\_Profit FROM WORK\_FACT\_TABLE

GROUP BY ROLLUP(Veh\_ID, Work\_Cost);

/\*Identify which maintenance actions on which vehicles are bringing in the most money\*/

SELECT Veh\_ID, MaintAct\_ID, Count(\*), Work\_Cost, SUM(Work\_Cost) AS Total\_Profit FROM WORK\_FACT\_TABLE

GROUP BY ROLLUP(Veh\_ID, MaintAct\_ID, Work\_Cost);

/\*Identify the amount of electrical and chassis parts that were used during December 2017, with cost.\*/

SELECT Parts\_Category, Work\_Cost, SUM(Work\_Cost), Count(Parts\_Category) AS Amount\_Used FROM WORK\_FACT\_TABLE f

INNER JOIN TIME\_TABLE t ON f.Time\_ID = t.Time\_ID

INNER JOIN PARTS p ON f.Parts\_ID = p.Parts\_ID

WHERE Time\_MonthNum = 12 AND Time\_Year = 2017 AND Parts\_Category IN ('Electrical','Chassis')

GROUP BY ROLLUP(Parts\_Category, Work\_Cost);

/\*Use the CUBE to determine the number of Maintenance Actions taken, along with number of individual hours worked and a sum of all hours for each maintenance action and overall.\*/

SELECT MaintAct\_ID, Count(MaintAct\_ID),

Work\_NumOfActHours AS Ind\_Action\_Hours\_Worked, SUM(Work\_NumOfActHours) AS Total\_Hours\_Worked

FROM WORK\_FACT\_TABLE

GROUP BY CUBE(MaintAct\_ID, Work\_NumOfActHours);

1. **Report Analysis**

/\*Scenario 1\*/

SELECT p.Parts\_Id, p.Parts\_Name, m.MaintAct\_Id, m.MaintAct\_Name

FROM Work\_Fact\_Table w

INNER JOIN Parts p ON w.Parts\_ID=p.Parts\_ID

INNER JOIN Maintenance\_Actions m ON w.MaintAct\_ID=m.MaintAct\_ID

WHERE w.MaintAct\_ID IN (SELECT MaintAct\_ID FROM

(SELECT MaintAct\_ID, COUNT(MaintAct\_ID) AS Maint\_Count

FROM Work\_Fact\_Table

GROUP BY MaintAct\_ID

ORDER BY Maint\_Count DESC)

WHERE Maint\_Count >= 5)

ORDER BY m.MaintAct\_ID;

/\*Scenario 2\*/

SELECT Time\_Month, Time\_Qtr, SUM(Work\_Cost) AS Cost\_Summary

FROM Work\_Fact\_Table w

INNER JOIN Time\_Table t ON w.Time\_ID=t.Time\_ID

WHERE Time\_Year = 2017

GROUP BY CUBE(Time\_Month, Time\_Qtr)

ORDER BY Time\_Qtr, Time\_Month;

/\*Scenario 3\*/

SELECT c.Cust\_ID, c.Cust\_FName, c.Cust\_LName, m.MaintAct\_ID, m.MaintAct\_Name

FROM Work\_Fact\_Table w

INNER JOIN Maintenance\_Actions m ON w.MaintAct\_ID = m.MaintAct\_ID

INNER JOIN Customers c ON w.Cust\_ID=c.Cust\_ID

WHERE m.MaintAct\_ID IN (

SELECT MaintAct\_ID FROM (

SELECT c.Cust\_ID, m.MaintAct\_ID, COUNT(w.MaintAct\_ID) AS Maint\_Count

FROM Work\_Fact\_Table w

INNER JOIN Maintenance\_Actions m ON w.MaintAct\_ID = m.MaintAct\_ID

INNER JOIN Customers c ON w.Cust\_ID=c.Cust\_ID

GROUP BY c.Cust\_ID, w.MaintAct\_ID, m.MaintAct\_ID

ORDER BY Maint\_Count DESC)

WHERE ROWNUM =1);

/\*Scenario 4\*/

SELECT w.MaintAct\_ID, m.MaintAct\_Name, SUM(w.Work\_Cost)

FROM Work\_Fact\_Table w

INNER JOIN Maintenance\_Actions m ON w.MaintAct\_ID=m.MaintAct\_ID

GROUP BY w.MaintAct\_ID, m.MaintAct\_Name

ORDER BY SUM(Work\_Cost) ASC;

References

Software Development Process – activities and steps. (2010). Retrieved from http://www.uacg.bg/filebank/acadstaff/userfiles/publ\_bg\_397\_SDP\_activities\_and\_steps. pdf

# The Benefits of Data Warehousing and ETL(April 5, 2016). Retrieved from https://www.glowtouch.com/blog/analytics/the-benefits-of-data-warehousing-and-etl/