Project 4

In this assignment you will complete the creation of your OVS, Inc. data warehouse database tables and then perform a number of decision support queries using the tables.

You can perform this assignment based on a database on Nova or any other Oracle system you wish, but you must use the Oracle RDBMS.

You should use one or more SQL script files to complete this assignment. Your script files should contain all your SQL and PL/SQL code. Do NOT submit your SQL script files. Doing so may result in confusion and will result in lost points.

Everything for this assignment must be in a single file. If you are using SQL*Plus you must put all your SQL, PL/SQL, and results together in a single SPOOL file. If you are using SQL Developer or other GUI, put all your screen snapshots in a single file for both your SQL statements and PL/SQL as they executed and the results. Failure to include all your SQL, PL/SQL, and all your Oracle execution results along with them will result in lost points.

Do NOT submit additional files as this only complicates the grading, and will result in lost points.

The specific assignment steps are listed below. In order to earn full credit you must keep your steps in the order shown, number your steps, and put everything in a single file.

1) Create the TIMES star schema dimension table via SQL. This is “TIME” table in the star schema figure. The Sale_Day primary key column values should be all dates from your first sale date through and including your last sale date from your SALES table. The Day_Type values should be 'Weekday', 'Weekend', or 'Holiday' (this trumps Weekday and Weekend). Set all occurrences of the following days for your date range to be holidays: New Year’s Day, Martin Luther King Jr’s Birthday, President’s Day, Memorial Day, 4th of July, Labor Day, Columbus Day, Veterans Day, Thanksgiving, and Christmas. Use a PL/SQL block to populate the TIMES table. After populating your TIMES table execute the SQL statement "SELECT day_type, COUNT(*),MIN(sale_day),MAX(sale_day) FROM time GROUP BY day_type ORDER BY day_type" to show the summarized contents of your table. Show all your SQL and PL/SQL code for this step and the results.

2) Create the SALES_FACTS star schema fact table via SQL. Ensure that you have declared foreign keys of Sale_Day, Vehicle_Code, Plan_Code, and Dealer_ID to reference your TIMES, VEHICLES, FINANCING_PLANS, and DEALERSHIPS tables, respectfully. Ensure that you have a primary key for the SALES_FACTS table that is a composite of the Sale_Day, Vehicle_Code, Plan_Code, and Dealer_ID columns. Do a DESC (i.e. DESCRIBE) of your SALES_FACTS table after it’s created.

3) Ensure that the FINANCING_PLANS, DEALERSHIPS, VEHICLES, and TIMES dimension tables are all created and populated as required. Do a SELECT COUNT(*) FROM <table_name> for each of the four tables.

4) Using PL/SQL stored procedure populate the SALES_FACTS table. One way to do this is to use four nested cursor loops to get every possible combination of the dimension tables’ primary keys and then the total vehicles sold and gross sales amount for each combination. If these values for

Total_Vehicles_Sold and Gross_Sales_Amount for a combination are zero then don't INSERT a row into the SALES_FACT table. Only insert rows for combinations of the four foreign key columns where there were some vehicles sold. Another approach besides nested cursor loops is to use a single INSERT statement with a GROUP BY clause. After populating your SALES_FACTS table execute the query "SELECT COUNT(*) FROM sales_facts;" to show the row count. Also execute the query "SELECT SUM(vehicles_sold) FROM sales_facts;" to ensure that you have included all of your 200 or more sales.

5) Create a user-defined function called VEHICLES_BY_VEHICLE_TYPE that receives an input parameter of a concatenated make and model and then queries the VEHICLES and SALES_FACTS tables to return the total vehicles sold by that combination. Execute your function for a sample input value of your choosing to demonstrate that it works correctly.

6) Create a user-defined function called DOLLARS_BY_VEHICLE_TYPE that receives an input parameter of a concatenated make and model and then queries the VEHICLES and SALES_FACTS tables to return the total gross sales amount of the sales by that combination. Execute your function for a sample input value of your choosing to demonstrate that it works correctly.

7) Create a stored procedure called STATS_BY_VEHICLE_TYPE that receives an input parameter of the concatenated make and model and then calls your two user-defined functions VEHICLES_BY_VEHICLE_TYPE and DOLLARS_BY_VEHICLE_TYPE. Your stored procedure must return the results of the two functions' executions via OUT parameters. Execute your stored procedure for the same sample input values used earlier to demonstrate that it works correctly.

8) Develop an SQL query to determine which holiday had the most sales and then drill down via another query to determine for that holiday which dealership had the most sales. Then drill down by another query to determine for that holiday which dealership, by zip code, and make had the most sales.

9) Develop an SQL query to determine how many days, by day type, didn't have more than 2 vehicles and $30,000 total sales. In order to achieve a fair comparison, analyze your results by the total number of weekdays, weekend days, and holidays in your TIMES table.

Your submission MUST be in a single text, Word, or PDF file with all steps numbered and in order.

Project 4 grading rubric

Attribute	Meets	Does not meet
CREATE TABLE SQL statements	**10 points** Uses an SQL script file. Creates the TIMES star schema dimension table via SQL. The Sale_Day primary key column values should be all dates from your first sale date through and including your last sale date from your SALES table.	**0 points** Does not use an SQL script file. Does not create the TIMES star schema dimension table via SQL. The Sale_Day primary key column values are not all dates from your first sale date through and including your last sale date from your SALES table.

	The Day_Type values should be 'Weekday', 'Weekend', or 'Holiday' (this trumps Weekday and Weekend). Sets all occurrences of the following days for your date range to be holidays: New Year's Day, Martin Luther King Jr's Birthday, President's Day, Memorial Day, 4th of July, Labor Day, Columbus Day, Veterans Day, Thanksgiving, and Christmas. Create the SALES_FACTS star schema fact table via SQL. Ensures that you have declared foreign keys of Sale_Day, Vehicle_Code, Plan_Code, and Dealer_ID to reference your TIMES, VEHICLES, FINANCING_PLANS, and DEALERSHIPS tables, respectfully. Ensures that you have a primary key for the SALES_FACTS table that is a composite of the Sale_Day, Vehicle_Code, Plan_Code, and Dealer_ID columns. Ensures that the FINANCING_PLANS, DEALERSHIPS, VEHICLES, and TIMES dimension tables and the SALES_FACTS fact table are all created and populated as required Includes all necessary integrity constraints including primary keys, foreign keys, CHECK constraints, UNIQUE constraints, and NOT NULL constraints. Uses an Oracle RDBMS. All SQL statements are syntactically correct and execute without error.	The Day_Type values are not 'Weekday', 'Weekend', or 'Holiday' (this trumps Weekday and Weekend). Does not set all occurrences of the following days for your date range to be holidays: New Year's Day, Martin Luther King Jr's Birthday, President's Day, Memorial Day, 4th of July, Labor Day, Columbus Day, Veterans Day, Thanksgiving, and Christmas. Does not create the SALES_FACTS star schema fact table via SQL. Does not ensure that you have declared foreign keys of Sale_Day, Vehicle_Code, Plan_Code, and Dealer_ID to reference your TIMES, VEHICLES, FINANCING_PLANS, and DEALERSHIPS tables, respectfully. Does not ensure that you have a primary key for the SALES_FACTS table that is a composite of the Sale_Day, Vehicle_Code, Plan_Code, and Dealer_ID columns. Does not ensure that the FINANCING_PLANS, DEALERSHIPS, VEHICLES, and TIMES dimension tables and the SALES_FACTS fact table are all created and populated as required Does not include all necessary integrity constraints including primary keys, foreign keys, CHECK constraints, UNIQUE constraints, and NOT NULL constraints. Does not use an Oracle RDBMS. All SQL statements are not syntactically correct or execute without error.
INSERT SQL statements	**10 points** All SQL statements are syntactically correct and execute without error.	**0 points** All SQL statements are not syntactically correct or execute without error.

SELECT SQL statements	**25 points** Develops an SQL query to determine which holiday had the most sales. Drills down via another query to determine for that holiday which dealership had the most sales. Drills down by another query to determine for that holiday which dealership, by zip code, and make had the most sales. Develops an SQL query to determine how many days, by day type, didn't have more than 2 vehicles and $30,000 total sales. Analyzes your results by the total number of weekdays, weekend days, and holidays in your TIMES table. After populating your TIMES table execute the SQL statement "SELECT day_type, COUNT(*),MIN(sale_day),MAX(sale_day) FROM time GROUP BY day_type ORDER BY day_type" to show the summarized contents of your table. All SQL statements are syntactically correct and execute without error.	**0 points** Does not develop an SQL query to determine which holiday had the most sales. Does not drill down via another query to determine for that holiday which dealership had the most sales. Does not drill down by another query to determine for that holiday which dealership, by zip code, and make had the most sales. Does not develop an SQL query to determine how many days, by day type, didn't have more than 2 vehicles and $30,000 total sales. Does not analyze your results by the total number of weekdays, weekend days, and holidays in your TIMES table. Does not execute the SQL statement "SELECT day_type, COUNT(*),MIN(sale_day),MAX(sale_day) FROM time GROUP BY day_type ORDER BY day_type" to show the summarized contents of your table. All SQL statements are not syntactically correct or execute without error.
PL/SQL stored procedure	**25 points** Uses a PL/SQL stored procedure populates the SALES_FACTS table. Creates a stored procedure called STATS_BY_VEHICLE_TYPE that receives an input parameter of the concatenated make and model and then calls your two user-defined functions VEHICLES_BY_VEHICLE_TYPE and DOLLARS_BY_VEHICLE_TYPE. Your stored procedure must return the results of the two functions' executions via OUT parameters. All PL/SQL code is syntactically correct and executes without error.	**0 points** Does not use a PL/SQL stored procedure to populate the SALES_FACTS table. Does not create a stored procedure called STATS_BY_VEHICLE_TYPE that receives an input parameter of the concatenated make and model and then calls your two user-defined functions VEHICLES_BY_VEHICLE_TYPE and DOLLARS_BY_VEHICLE_TYPE. Your stored procedure must return the results of the two functions' executions via OUT parameters. All PL/SQL code is not syntactically correct or executes without error.

PL/SQL user-defined functions	**25 points** Creates a user-defined function called VEHICLES_BY_VEHICLE_TYPE that receives an input parameter of a concatenated make and model and then queries the VEHICLES and SALES_FACTS tables to return the total vehicles sold by that combination. All PL/SQL code is syntactically correct and executes without error.	**0 points** Does not create a user-defined function called VEHICLES_BY_VEHICLE_TYPE that receives an input parameter of a concatenated make and model and then queries the VEHICLES and SALES_FACTS tables to return the total vehicles sold by that combination. All PL/SQL code is not syntactically correct or executes without error.
SQL script file and SPOOL file	**5 points** Does a DESC (i.e. DESCRIBE) of your SALES_FACTS table after it's created. Shows all your SQL and PL/SQL code for each step and the results. Demonstrates full ability to create and use an Oracle SQL script file and output SPOOL file.	**0 points** Does not perform a DESC (i.e. DESCRIBE) of your SALES_FACTS table after it's created. Does not show all your SQL and PL/SQL code for each step and the results. Does not demonstrate full ability to create and use an Oracle SQL script file and output SPOOL file.