Sales planning and evaluation

Staff graded assignment

Week 6

12 points

Contents

1. Building an IT-solution to perform a plan-fact analysis of bike sales

1.1. Description

The main learning objective of this assignment is to apply knowledge of SQL and foundations of relational databases to create a simple IT-solution for plan data management.

You have been given access to a database of a company that sells bikes and related products. The database contains information about orders, products, and customers. Customers of two types place orders - individuals and shops (reseller companies).

To evaluate the performance of sales department you need to (1) create tools to automate the planning process and (2) prepare data to calculate difference between planned and actual values of amount sold to companies.
Data manipulations should be implemented in SQL. Plan management (like creating new plan periods, data approval or changing planning status) needs to be done in form of python function calls.

This assignment includes the following activities:

- Writing queries to process and copy data
- Creating functions in python that uses one or more SQL queries
- Creating materialized views in SQL
- Setting permissions with SQL

A more detailed explanation of planning process is presented in the sections below. The next chapter covers users of a planning system you are creating and their role in planning process.

1.2. Users of planning system

There are two user groups working with plan data: administrators and managers.

Administrators prepare initial plan data which then becomes available to managers. One planning period stands for one quarter. Administrators have access to all plan data unlike managers.

Managers plan sales in several countries. Mapping between countries and managers is documented in the system's settings. Once the administrator has prepared the initial version of sales plan, managers will be able to update and approve the calculated figures. The initial sales plan is based on actual data in two previous years.

Managers can lock their data to avoid unintended corrections by other users. Only locked data is available for correction.

At the end of the planning process managers confirm correctness of plan figures. The approved plan is then used to prepare a report on plan-fact deviations.

Plan-fact comparison report is available to both managers and administrators.

1.3. Implementation of the planning process in a database

To organize planning process special tables were designed to store planning data as well as information on its status on a given planning period.

Plan data has three versions stored in a single table - N (initial calculation), P (edited) and A (confirmed and considered as the result of planning).

A plan data slice is defined by a quarter (e.g., 2013.3) and a country. The status of each data slice can be set individually. The valid status values are R (ready for work), L (locked by manager and available for editing), A (approved).

This list contains an example of operations performed on plan data:

- The planning period (quarter and year) is defined
- The administrator creates current planning period records with an R status for each slice of plan data. If there are 6 countries, then 6 records will be added.
- Initial plan calculation is performed, and the results are saved as an 'N' version. Version N is not changed afterwards.
- Plan data is copied from N to P version. The only difference between these versions of a plan is only the version name
- Managers change P version of plan data after changing the respective slice status from R to L. If the data was locked by another person before, then editing is impossible.
- Managers release locks on their data. Removing data locks means changing data status from L to R. The lock can be removed only if the slice status is L and the author of last changes is the current user.
- Then the data is copied to the 'A' version by managers which means that the plan data is approved. The respective status records of data slices are changed from R to A. Data in the 'A' version is then used to prepare a report on differences between planed and actual sales amount.

1.4. Calculating the initial planning values

To evaluate the expected volume of sales to shops, you need to use actual sales to customers, which make the most of total yearly sales to companies (not individuals).

Company (shop) grouping is conducted with help of ABC-analysis based on sales data of each year. Only A and B groups of customers are used to prepare the initial plan data.

The planning year is further denoted as ***y***.

To determine the initial target sales amount, it is necessary to calculate the quarterly sales amounts in (***y – 1)*** and (***y – 2)*** years (excluding customers in the group C) based on planning period, quarter and year of fact data, country and product category. Then calculate quarterly averages for each combination of planning period, country and product category.

The administrator saves the initial data as an 'N' version of the plan. The same data is copied to the 'P' version which is supposed to be edited by the managers.

For example, the calculation in the first quarter of 2020 for Bikes in US will require data in first quarter of 2018 (2018Q1_US_Bikes) and the first quarter of 2019 (2019Q1_US_Bikes). The result of calculations will be equal to (2018Q1_US_Bikes + 2019Q1_US_Bikes)/2.

If there were no sales in one of the two years, then do not divide the number by 2.

1.5. Company classification by annual orders

The assessment is done quarterly in terms of product categories and the countries in which the stores are located. In this task the planning year is 2014.

It is necessary to split the buyer companies into three groups by their volume of orders in ***y - 1*** (2013) and ***y - 2*** (2012) years. As a result, two ratings of companies (for each year) will be created. The following algorithm should be used to define members of the three groups for one year:

1. Calculate the total volume of sales to companies before taxes. Denote the value as S
2. Calculate the upper boundary values for inclusion in the groups A and B:
   $S_a = 0.8 * S$,
   $S_b = 0.95 * S$
3. Sort companies in descending order by the total volume of their respective purchases made per year (ST_i).
4. Calculate a running total of ST_i and denote these values as SRT_i. SRT_i of a single company should be equal to the sum of the accumulated purchase volume made by this company per year and the aggregated purchase volumes made by all the companies that are higher in the rating.
5. Assign a class to eacht company (*i*) according to the following conditions:
   a. Class A – if SRT_i does not exceed S_a,
   b. Class B – if SRT_i does not exceed S_b,
   c. Class C – in any other case.

1.6. Comparison of planned and actual sales

As soon as a manager has finished editing the plan and approved it, the data becomes available for plan-fact analysis.

The plan-fact analysis report uses only those plan values that are marked as approved by the manager.

Comparison of planned and actual values is made in terms of year, quarter, country, and product category. Actual figures are taken from the information about sales in the ***y*** year to shops that were members of groups A or B in the ***y – 1*** year.

If the planning period is the 1st quarter of 2020, then the actual data will be selected from the 1st quarter of 2020 for companies that were in groups A or B as of 2019.

A report template is shown below:

Table 1

Field	**Quarter**	**Country**	**Category name**	**Dev.**	**Dev., %**
Number	1	2	3	4	5
Description	Quarter key in YYYY.Q format	Code of country	Category name	=Plan – Fact Plan is chosen from the A version of plan data. Fact is from fact data (linetotal)	=(Plan-Fact)/Plan

Examples	2014.1, 2014.2, ...	AU, DE, …	Bikes,…	10000, -3000,…	10%, -30%, …

Table 1

If the approved plan data cannot be found, then null should be shown in cells 4 and 5.

2. About this task

2.1. Instructions

While preparing this assignment you will modify access settings and write SQL code to manage plan data and create additional database objects. Then you will develop several Python functions to automate activities related to the manipulation with plan data.

The code written will be used to prepare a plan for the first quarter of 2014. The final step includes comparing planned and actual sales figures.

2.2. How to submit

To get mark for this assignment you need to prepare two files and upload them to the platform.

1. Create a report in docx format. Every task includes a hint on what should be put in the report.

For example:

Add 'Confirmation of Plan Data' to the report. Insert the function code into this chapter.

2. After you are done with the whole assignment generate your database dump file with pg_dump and submit it with the report.

2.3. How this assignment will be graded?

This assignment has 9 tasks. Instructions begin on page 8.

First, the completeness of the report is considered. To get a high score, all results marked with *color and italics* must be present in the report and well formed.

The backup should include all the results mentioned in the report. Code from the report should align with requirements written in this document.

Develop functions and queries that are compatible with the database.

You get 1 point for tasks 1, 2, 3, 5, 7, 8 and 2 points for 4, 6, 9.

3. Before you begin

Make sure, you have installed Postgres 11 or later. You will also need a client software to communicate with the server (for example, DBeaver).

Find files with the following titles (attached to this course):

- FT. Database backup - backup database with business data
- FT. Tables, views and roles - DDL to create additional tables, views and roles

Restore the database on your server and name it like <year>_plans_<your name>. E.g. 2020_plans_Kirill. In case of success your database will contain tables as shown on the diagram below (Figure , page 7).

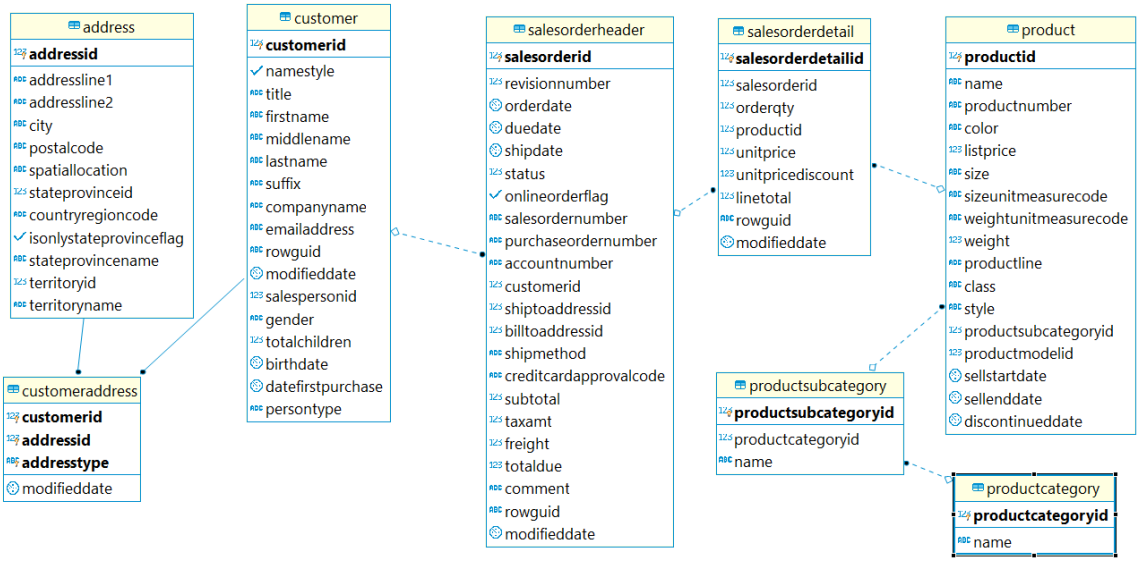

Figure 1 Tables with business data

In addition to the tables in the diagram, you will need tables and views below, which should be added from the corresponding file with DDL. Run code from the file to create the following database objects:

1. Table which stores status of plan data fragments – 'plan_status'
2. Table of versioned plan data – 'plan_data'
3. Data access settings defining each manager's permissions to view and edit sales plans in specific countries – 'country_managers'
4. Table of customers contains records about individuals and shops that resell products from the firm you are creating a database project for. Shops are also referred to as companies or customers in this assignment.
5. Aggregated sales table in terms of companies, product categories, years, and quarters – 'fact_sales'
6. Company classification by total annual cost of orders – 'company_abc'
7. A view for managers to edit planned data – 'v_plan_edit'
8. A view to read approved planning data based on user's permissions – 'v_plan'
9. Two roles – planadmin (administrators) and planmanager (managers).

4. Graded tasks

4.1. Task №1. Access settings

Set up permissions for roles as written in the table below:

Table 2 Planning system users' rights

	User role	
DB object	planadmin	planmanager
all tables	S	S
plan_data	SUID	SUID
plan_status	SUID	SU
country_managers	SUID	S
v_plan_edit	S	SU
v_plan	S	S

Legend:
S – select
U – update
I – insert
D - delete

Create three users:

- Administrator:
  - ivan
- Managers:
  - sophie
  - kirill

Sophie has access to data related to US and CA countries. Kirill works with sales data in FR, GB, DE, AU countries. Put this information in the 'country_managers' table, which will associate managers with certain countries they are responsible for.

Populate the 'country_managers' table with the records on two managers – sophie and kirill – having sophie responsible for US and CA, and kirill – for FR, GB, DE, AU. Insert a query/queries into the report to put the data into the 'country_managers' table.

4.2. Task №2. Creating product and country views

Add two materialized views – 'product2', 'country2'. Product2 should combine data of product and its category. Country2 view should be filled with unique codes of the countries where the shops are located (country codes can be taken from addresses of 'Main Office' type).

Allow managers and administrators to read from these views.

Fields of 'product2' are shown in the table. Each field to be included in the view is described below:

Table 3 product2 materialized view

Field	Description	Rule
pcid	Product category key	Load from Productcategory.productcategoryid
productid	Product key	Load from Product.productid
pcname	Category name	Load from Productcategory.name
pname	Product name	Load from Product.name

Fields of 'country2' are shown in the table:

Table 4 country2 materialized view

Field	Description	Rule
countrycode	Code of country	Load unique values from customeraddress.countreyregioncode Make sure your query reads only 'Main Office' addresses

Add sql code of the views into the report under 'Task №2. product2 & country 2 materialized views' heading. Also add commands to set necessary permissions.

Tables with source data

Diagrams of needed tables are presented below. You can use the highlighted columns to connect the tables.

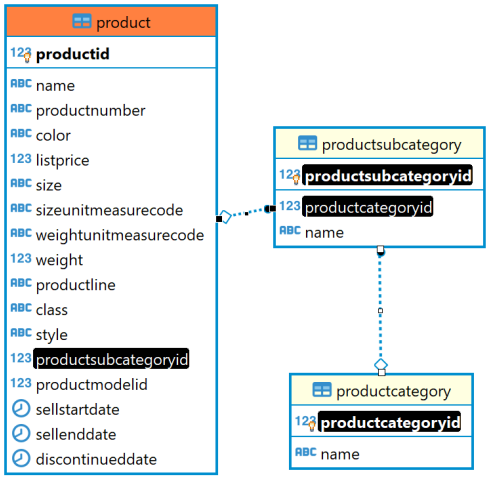

Figure 2 Tables for product2

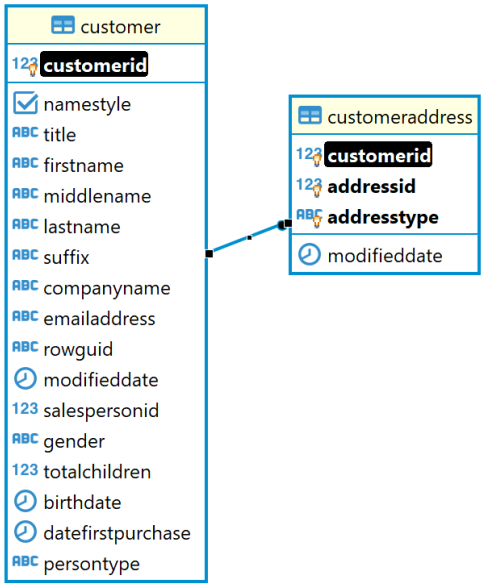

Figure 3 Tables for country2

4.3. Task №3. Loading data into the company table

In the current database the *customer* table contains information about two categories of buyers - individuals and companies. However, we consider only companies. For the convenience of further development, fill the company table with data.

Data from the companyname field should be included in the list of companies. The country and the city should be taken from the address table. Develop a query to load the country table.

Follow these rules in the table below. Use an addresses of 'Main Office' type to find the appropriate information.

Table 5 'company' table fields

Field	Description	Rule
id	Company key	Auto-generated value. No need to populate it manually
cname	Company name	Company name 'customer.companyname'
countrycode	Code of country	Head office country 'address.countryregioncode'
city	City name	Head office city 'address.city'

Include the prepared query into report under the heading 'Task №3. Loading data into the company table'

Tables with source data

The following tables can be used to form the 'company' table: customer, customeraddress, address. Find them on the diagram.

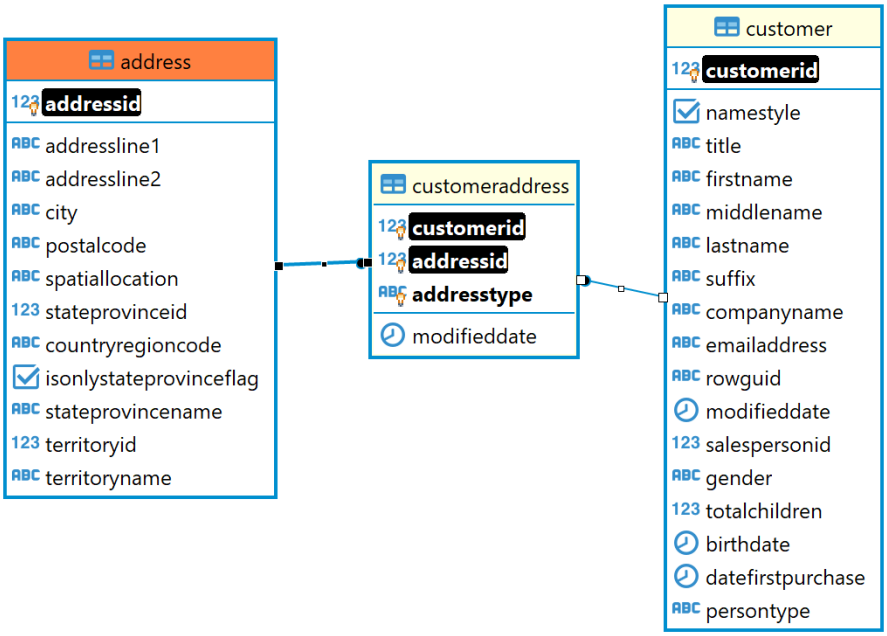

Figure 4 Source data to fill 'company' table

4.4. Task №4. Company classification by annual amount of purchases

Split the companies into three groups according to algorithm in section 1.5 (on page 5) for 2012 and 2013.

Populate the 'company_abc' table after grouping the companies. All calculations should be done in one query. The detailed calculation rules are given below.

Table 6 Calculation rules for company_abc table

Field	Description	Rule
cid	Company key	Key from 'company' table. Find the *companyname* from '*customer*' in the 'company' table to retrieve its key
salestotal	Sales total per year	Calculated as *sum(subtotal)* from *'salesoderheader'* table
cls	Company class	The class should be calculated following an algorithm in the section 1.5.
year	Year	Get year of *orderdate* field from *'salesoderheader'* table

Add the designed SQL queries to the report under 'Task №4. Company classification' heading. Add a screenshot of 10-20 records of company_abc.

Tables with source data

Use the following tables:

salesorderheader, customer, company

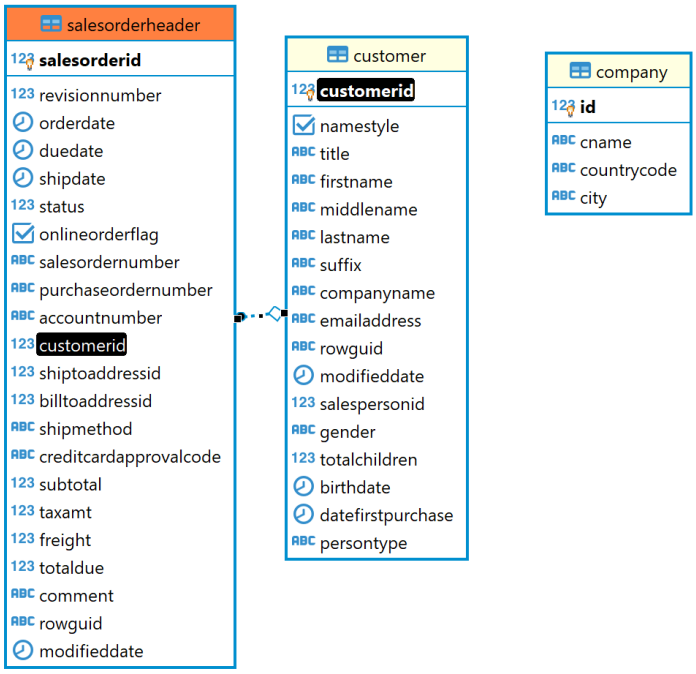

Figure 5 Tables for company segmentation

4.5. Task №5. Finding quarterly volume of purchases made by each company, and the product category

Calculate quarterly sales amount before taxes in 2012 and 2013 individually. Fill the *company_sales* table using data about orders, companies, and classification results in the respective year.

The table below contains comments on filling the fields of the *company_sales* table.

Table 7 Data loading rules for company_sales table

Field	Description	Rules
cid	Company key	*company.id* Find *company.id* through *customer* and *company* tables
salesamt	Total amount sold (before taxes). Purchase volume of each product	Calculate as sum(*salesorderdetail.linetotal*) for combinations of

	category made per quarter by each company	year&quarter, company, and category. The *salesorderdetail* table contains product sales data in quantity and cash.
year	Numeric year of orderDate	Numeric year of *salesorderheader.orderdate*
quarter_yr	Numeric quarter number of orderDate	Numeric quarter number of *salesorderheader.orderdate*
qr	Quarter in text format. E.g., *2022.1* or *2023.4*	String representation of quarter in form of YYYY.Q extracted from *salesorderheader.orderdate.*
categoryid	Product category's key	*product2.pcid.* Connect *product2* with *salesorderdetail*
ccls	Company's class name. E.g., A, B or C.	String name of a class to which a company belongs. Use *company_abc.cls* for the year from *salesorderheader.orderdate*

Add your query into the report under "Task №5. Finding quarterly sales amount by company, and product category" heading

Source data

Use the following tables and views to complete the task:

Customer, company, company_abc, salesorderheader, salesorderdetail, product2

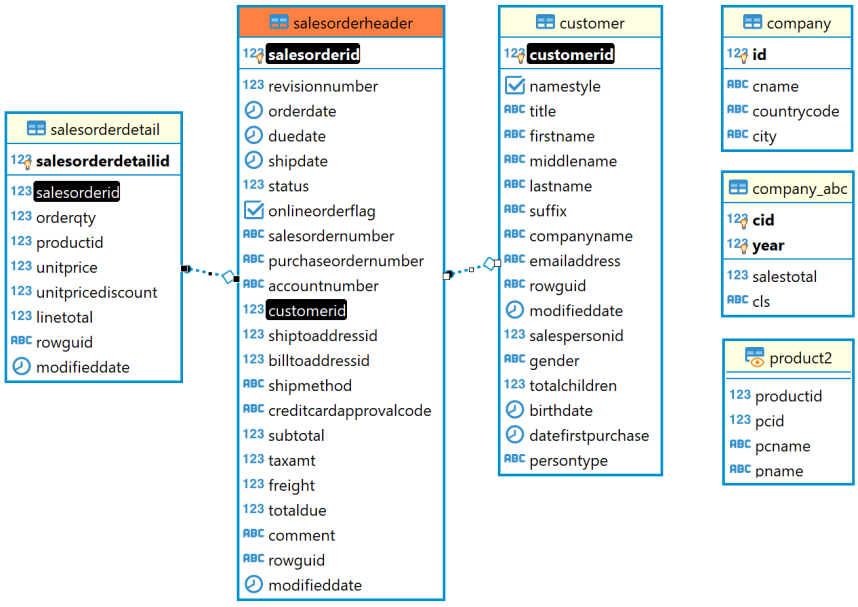

Figure 6 Source tables in the database

4.6. Writing functions to automate planning process

4.6.1. Task №6. Generating the initial planning data

Write a function in python to help managers start working with plan data.

You can call *con*.commit() at the end of processing to apply changes to the data after you execute SQL commands. The *con* variable here is a connection in *psycopg2* library. If you do not add this instruction, then the data will be left unmodified.

Write a function start_planning(year, quarter, user, pwd) where:

- year is the target year of the planning period,
- quarter is the target quarter of the planning period,
- user is the username of the database user,
- pwd is this user's password

These parameters have the same meaning in the next tasks.

The function should implement the following steps:

1. Delete plan data from the *plan_data* table related to the target year and quarter
2. Delete all records related to the target quarter from the *plan_status* table
3. Create the necessary planning status records in the *plan_status* table for the selected quarter. The number of records added should be equal to the number of countries in which customer-companies (shops) are situated.
4. Generate a version N of planning data in the *plan_data* table. Use the calculation algorithm as described in section 1.4.
5. Copy the data from the version N and insert it further to the *plan_data* table changing the version to P
6. Store the name of the user who called the function, in the records of the *plan_status* table

If initial planning data cannot be generated for some record in *plan_status* (e.g., if no data can be found in the *company_sales* table), then add rows with 0 in *salesamt* column for related combination of country and category.

If the function works correctly, the data will appear in the *plan_data* and *plan_status* tables. The N and P versions of plan data will be created.

Populate the plan_status table with the following:

Table 8

Column	**Description**	**Rules**
quarterid	Key of planning quarter	String identifier of a quarter with "YYYY.Q" value template
country	Country of a company	*company.country*
status	Planning status	R
modifieddatetime	Time when the record was changed or created	current_timestamp
author	User who changed the record	current_user

Rules for loading data into the *plan_data* tables are shown here:

Table 9

Column	Description	Rules
versionid	Version of plan	a version of the plan: initial data generation gets N, a copy from N gets P
country	Country of a shop which orders goods	*country2.countrycode* or *company.countrycode*
quarterid	Key of planning quarter	String identifier of a quarter with "YYYY.Q" value template
pcid	Product category's key	*product2.pcid*
salesamt	Sales amount before taxes	average of total quarterly purchase volume in A and B groups calculated based on *company_sales.salesamt*

Call this function on behalf of Ivan. The target planning period is 1st quarter of 2014.

Add the start_planning function to the report under a new header - "Task №6. Initial data preparation". Write a line with function call that you used to populate the plan_data and plan_status tables. Add two screenshots of plan_data and plan_status contents, showing results of the function execution (P and N versions of plan should exist, status should be equal to R).

Source data

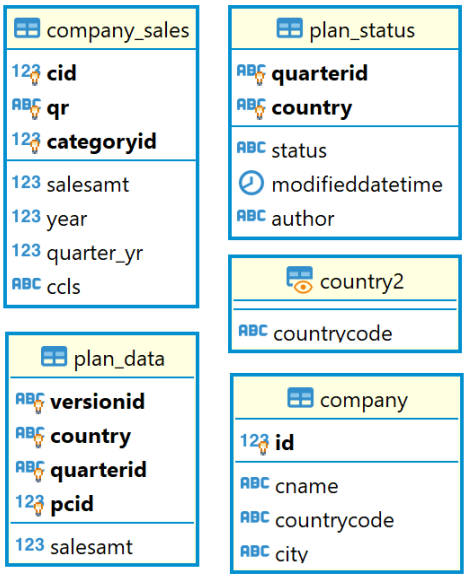

Figure 7 Tables with source data

4.6.2. Task №7. Changing the plan data

Write two functions in python:

- *set_lock*(year, quarter, user, pwd), which will change status from R to L for records in *plan_status*, that are associated with the target quarter and year, and connected to the current

user in the *country_managers* configuration table. To obtain the name of the current user, use current_user. Also write a timestamp of modification to the *modifieddatetime* field.

- remove_lock(year, quarter, user, pwd) function, that will change the planning data status from L to R. associated with the current user through the country_managers table. Also update the *modifieddatetime* field.

Execute the set_lock function to lock the plan data for the 1st quarter of 2014 on behalf of Kirill user, and then – Sophie. If everything is done correctly, data will appear in the *v_plan_edit* view if connected as Kirill or Sophie.

Increase the planned sales volume per country per category for the plan period by about 30-50% in the v_plan_edit view on behalf of two managers. You can edit data through the view in DBeaver using a virtual key (it must contain all fields except salesamt).

Run the function *remove_lock* to mark Q1 2014 as not in use. Run this function as Kirill and then as Sophie. Now the *v_plan_edit* view will return no records.

Add set_lock and remove_lock code into your report under "Changing plan data" header. Also provide a screenshot of v_plan_edit contents when logged in as kirill. The screenshot should show the changed data before executing the remove_lock function.

Rules for updating the *plan_status* table are listed below:

Table 10 Update rules for plan_status

Column	Description	Rules
quarterid	Key of planning quarter	No changes
sountry	Country of a shop which orders goods	No changes
status	Plan data slice status	L
modifieddatetime	Time when the record was changed or created	Value of current_timestamp
author	User that changed the record	Value of current_user

Tables used

Tables are shown on the picture below. Plan_status table should be updated. Country_managers stores information about users' permissions regarding plan sales to different countries.

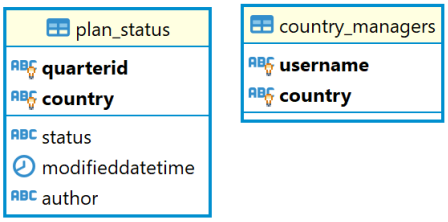

Figure 8 Tables to use inside functions that manage planning status

4.6.3. Task №8. Plan data approval

Write a function in python - *accept_plan*(year, quarter, user, pwd). The function will copy the modified data into the actual version of the plan.
The function is expected to select the records from the existent version of plan_data table, change the *versionid* there and then change the status in the *plan_status* table when the records in the original versions (P and R respectively) meet the following requirements:

- Planning quarter (*quarterid* column) is equal to combination of *year* and *quarter* from the function's arguments.
- Version equals to 'P' (which refers to a modified version of the plan).
- Data status (in *plan_status* table) equals to 'R'.
- The current user has a permission to access the plan data according to the settings in the *country_managers* table.

Implement these processing steps in the *accept_plan* function:

- Clear the A version of plan data for specific quarter and countries accessible to the current user
- Read data available to the current user from the version P and save its copy as the version A
- Change the status of the processed from 'R' to 'A'
- When updating the status, also save a timestamp in the *modifiedtimestamp* column.

Thus, after the function is applied, the tables should get their *versionid* and status updated to A as well as the author and modifieddatetime columns in *plan_status* should show the info of who and when made the latest changes.

Use the developed function to approve the plan of Q1 2014 on behalf of each manager. Check whether the data is visible through the *v_plan* view:

- The administrator has access to the entire plan
- Manager can view the only data he/she is permitted to read and change.

Add accept_plan function code to the report under "Plan data approval" heading. Also include a function call as kirill and sophie. After logging in as sophie add a screenshot of rows in the v_plan view.

Details on how each column can be populated are shown in the tables below.

Table 11 Rules for changing records in the plan_status table

Field	**Description**	**Rules**
quarterid	Key of planning quarter	No Changes
country	Country of a company	No Changes
status	Planning data status	'A'
modifieddatetime	Time when the record was changed or created	current_timestamp
author	User who changed the record	current_user

Table 12 Rules for loading data in the plan_data table

Field	**Description**	**Rules**
versionid	Version of plan	'A'
country	Country of a shop which orders goods	Copy from version P
quarterid	Key of planning quarter	Copy from version P
pcid	Product category's key	Copy from version P
salesamt	Sales amount before taxes	Copy from version P

Tables used

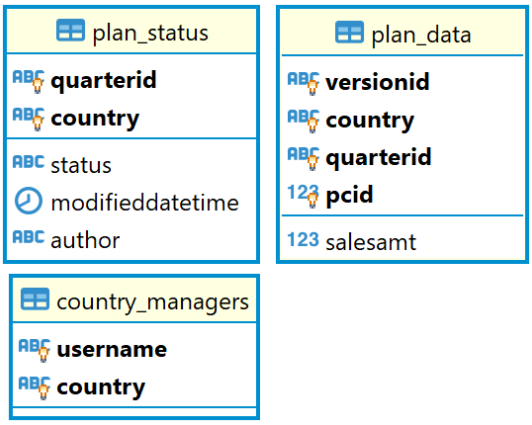

Figure 9 Tables for accept_plan(y, q, u, p)

4.7. Task №9. Comparison between the planned and actual sales in Q1 2014

Create a materialized view mv_plan_fact_2014_q1 to compare planned and observed sales before taxes in 1st quarter of 2014. The view itself should show the difference between fact and plan.

The requirements for the report are described in the section 1.6.

Use the classification results from 2013 to find actual sales of A- and B-class companies in 2014.

You can choose one of these options (does not affect your grade):

1. Load data of 2014 into the *company_sales* table and include this table in the view
2. Calculate actual data using *salesorderheader* and *ordersalesdetail* tables without using *company_sales*.

Add a header "Data preparation for plan-fact analysis in Q1 2014". Write which approach you chose. Include SQL code of the new materialized view together with a screenshot showing data in mv_plan_fact_2014_q1 view.

Tables used

Here you can find all tables and views that can serve as a data source for the considered report.

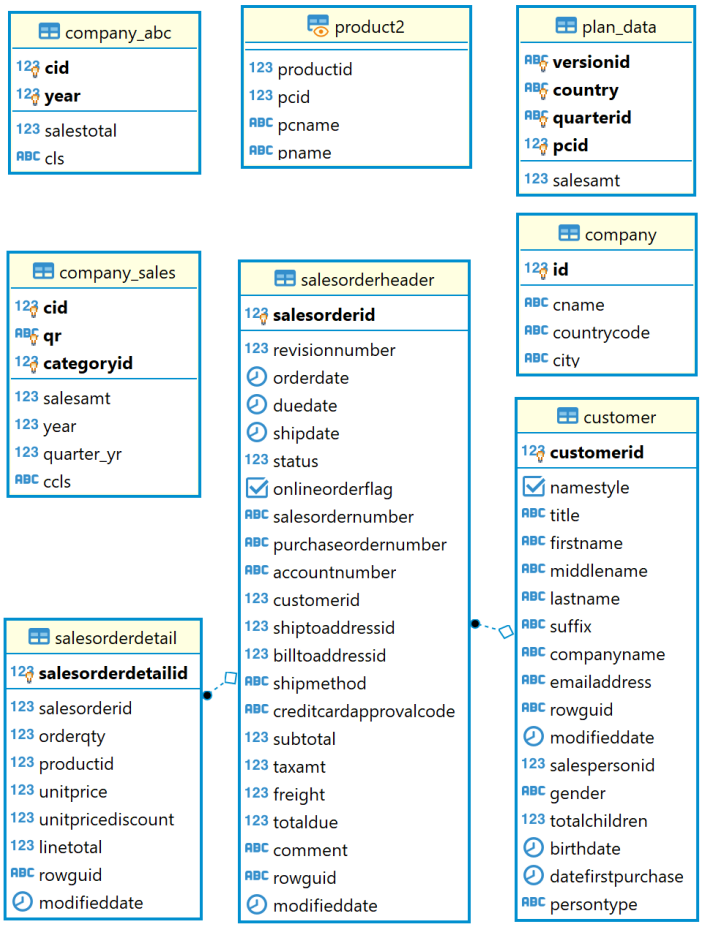

Figure 10 Tables and views available for plan-fact analysis

Database submission

Use backup instruments to prepare a dump file.

Attach the dump file and the report to the assignment's prompts. The database should meet the requirements described in this assignment – contain necessary tables with data and views.

Python functions and SQL queries provided in the report are expected to run successfully with your database.