UCONN
SCHOOL OF BUSINESS

UNIVERSITY OF CONNECTICUT SCHOOL OF BUSINESS
COURSE OPIM 5272 – FALL 2015 – CLASS PROJECT

THE TRAVEL APPROVAL & REIMBURSEMENT PROCESS

PROJECT PHASE I – PROCESS MODELING
PROJECT PHASE II – DATABASE DESIGN
PROJECT PHASE III – DATABASE IMPLEMENTATION

WRITTEN BY TEAM FOUR
RITTHA ARAYARUNGSARIT, DIE CHEN, ZHONG HE,
PRADEEP JEYACHANDRAN, JULIAN VROHLINGS

TABLE OF CONTENTS

The first part of this document describes a company, a process that needs improvement and a way to improve it. The second part focuses on the design of a database that supports this business process. The third part implements the database and delivers a series of SQL queries that can be used to generate insights from it.

PROCESS MODELING

The business – Stark Sealing Solutions

Our team has studied a simple business process in a medium-sized German company that has grown exceptionally in size within the last five years. Stark Sealing Solutions (real name replaced) offers all kinds of sealing solutions to other businesses. For example, solutions are offered to seal pipes, undersea fibre cables, windows, fuel tanks, tunnels and large machines. Due to the individual requirements and technical specifications, most employees work in small project teams together with the client. In a combined effort, they develop, create and test materials that are safe and durable for the product they shall be used for. As this usually takes place at the client site, it is important that employees are willing and able to travel within all of Europe. The company has 5,000 employees at three sites in Germany, France and Sweden, 3,000 of which need to travel for this reason on a regular basis.

Recently, there has been a growing number of complaints by both employees and managers, regarding the business rules and processes that have been set up around the topic of traveling. They say, it takes too much time to obtain approval for a trip to the customer, and also, it takes up to a month to get the money back (as most other companies, Stark asks its employees to advance the money spent on business travel and reimburse them later, as it speeds up booking/cancelation process). Additionally, when repeating the trip multiple weeks in a row, the process has to be run through every single time. A few years ago, when the company was much smaller, and most of its clients where located in the same region, the existing process may have been efficient, but as this has changed and traveling is very costly, and working time is a valuable resource, this is a process that needs to be examined.

The current process – sequential and paper-based

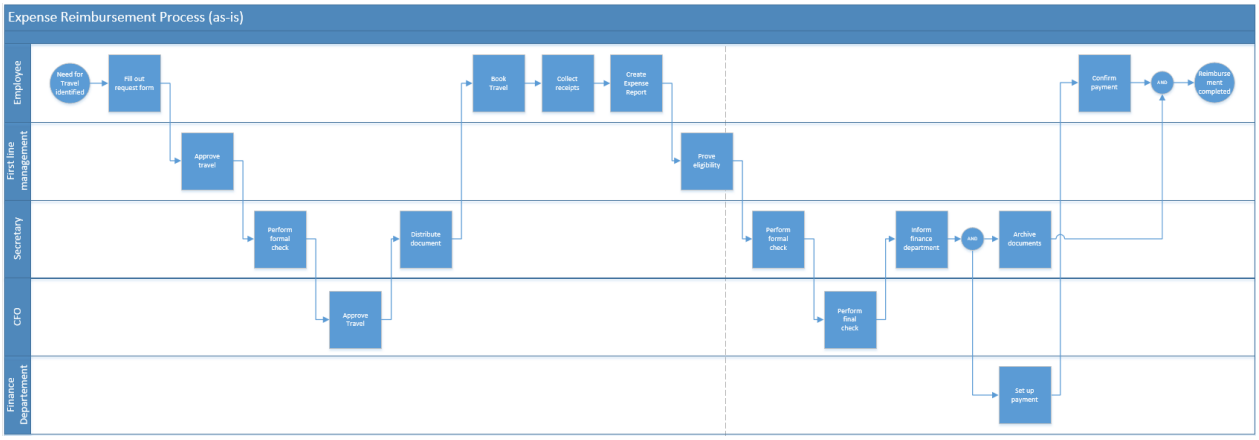
For compliance reasons, there are five different people involved all the way from identifying the need to travel (triggering event) and the reimbursement being completed. These people are:

- The employee requesting and booking a trip, traveling, collecting the receipts, creating the expense report, and finally confirming payment (responsible for achieving business success in projects)
- The employees first line manager, who has to perform a check if the employee is eligible to travel and if the expenditure matches the amount requested and the companies guidelines (responsible to ensure business success on all projects of his department)
- The CFOs secretary, who performs a formal check before giving it to her manager
- The CFO, who is the only one to make/sign decisions on spending
- The finance department, which can trigger the reimbursement to the employee

As the company handles and archives travel requests and reimbursement separately at each location, it has become common practice to print out and hand over everything to the next person. This is the reason why the process has to be sequential. Most of the time, the employee, and sometimes his manager identifies the need to travel to the client location (triggering event). After this, the process starts:

- The employee fills out a travel request form. He has to provide is name, workplace location, employee ID, department, the project and client ID, the estimated travel time and dates, estimated hotel, transportation and other costs, as well as a short description of the purpose of the travel.
    - He places the request on his managers desk or gives it to him in person
- The manager reads the form
    - No approval: he writes remarks on the request, instructing him how to adjust the request
    - Approval: he signs the request and places it on the CFOs secretaries desk
- The secretary double-checks if information is missing
    - Insufficient information: remarks on the request, including instructions on how to proceed
    - Sufficient information: collects forms from different employees and hands them over to the CFO for approval once a day
- The CFO reads through the request
    - No approval: he writes remarks on the request, including instructions for the secretary
    - Approval: he signs the request, handing it back to the secretary
- The secretary places the signed request form on the desk of the employee
- The employee proceeds with the booking. He is instructed to book only hotels, airlines, train services and rental car companies that are in from a “Partners of Stark“ list, if possible.
- The employee collects all receipts and bills while traveling
- After traveling, the employee creates the expense report. This includes an overview of the cost, using an excel form, printing it out and stapling all receipts to it
    - He places the report on his managers desk together with the initial request
- The manager reads the report and compares it the initial request
    - No approval: he writes remarks on the report, instructing him how to adjust it
    - Approval: he signs the report and places it on the CFOs secretaries desk
- The secretary double-checks if information is missing
    - Insufficient information: remarks on the report, including instructions on how to proceed
    - Sufficient information: collects reports/initial request from different employees and hands them over to the CFO for approval once a day
- The CFO reads through the report
    - No approval: he writes remarks on the report, including instructions for the secretary
    - Approval: he signs the report, handing it back to the secretary
- The secretary archives the report
- The secretary instructs the finance department (email including the amount to reimburse, the employee and project ID)
- The finance department transfers the money to the employees account
    - The finance department asks the employee for confirmation the money has arrived
- The employee confirms that he has received the money
- The reimbursement process is completed

This is a business process because it includes interrelated tasks (no task is independent from all other tasks), all tasks work towards a specific goal/result (reimburse employees), and it has a triggering event (in this case an action event). Compared to other business processes, this process is relatively easy, as all tasks are handled sequentially and the steps performed do not require deep understanding of finance. Nevertheless, as pointed out earlier, it is very important that business processes like these do not prevent employees from working efficiently on the main tasks they by taking away their time. Indirectly, the customer will benefit from improving this process. Starks project members should be less overwhelmed by the amount of administrative work to be done, and rather focus on the work to be done with the client. The direct „customers“ (or stakeholders) of this process are all internal: all parties involved in the as-is process described above would benefit from a streamlined travel and reimbursement process.



The current process – sequential and paper-based (to keep it simple, iterations have been removed)

Problems with the current process

As already pointed out, the overall process uses too much of the employees and managers time. This leads to very high risk and quite a few other inefficiencies:

- Managers can be sick/traveling/on vacation, the delay can prevent the employee from traveling (request has not been signed on time, hotels/rental cars/flights/trains are sold out in the meanwhile). This influences customer satisfaction directly in a very negative way.
- Most employees have enough money to advance a payment for a single trip. But if it takes up to a month to get the money back, this might be a major concern to them, prevent them from traveling or even make them wanting to leave the company if no action is taken.
- If there was no delay at handover, the booking part of the process could be finished in less than twenty minutes, the reimbursement process in less than half an hour. However, it is estimated that requests and reports are waiting for approval more than 95% of the time during the whole process (for example because the CFO only handles requests/expense reports once a day)
- Because it is paper-based, it is difficult to get a history of an employees trips, discover potential fraudulent activities, get a status of an open request/report or reduce the risk of error when comparing the request to the final expense report
- Because the process doesn’t work anymore, employees invent workarounds that sometimes are not compliant with business rules & legislation
- Project managers struggle to locate and assign travel-related costs to their projects, they must go to the archive or ask the people who traveled regularly

It is important to change the status quo, and enable Stark Sealing Solution to spend time on what creates revenue and customer satisfaction. This is not only true in the short term, but also in the long term, as the company continues to grow in size.

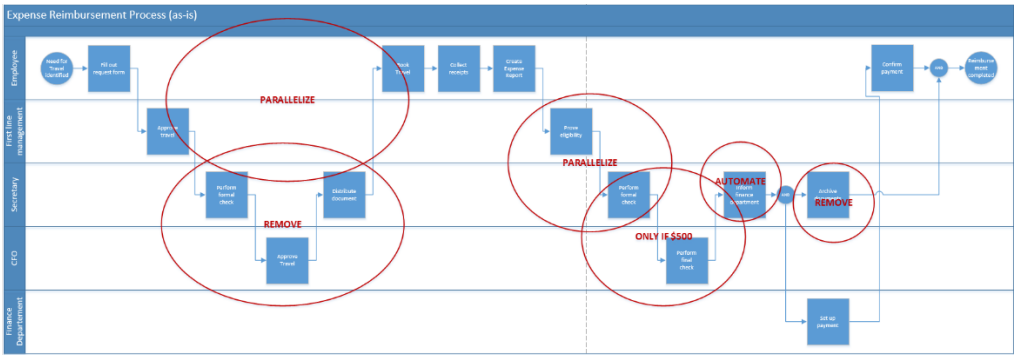
Improving the process

We have identified different areas in which the process can be changed in order to be more efficient:

- Workflow Redesign
  - Let the CFO delegate the travel request approval completely to the first line manager. This requires trust, but after all, the first manager is to be held accountable for every decision he makes and risks his own job if he spends money in a wasteful way. The CFO still has to approve the expense report, so he is still in control of the result.
  - Let the CFO only deal with expense reports greater than $500. For the same reason as posted above, the first line manager can in charge of minor decisions.
  - Parallelize booking and approval. A majority of requests is reasonable and will be approved anyway. The etiquette of waiting for approval before taking action can be removed. The employee may already book, if free cancelation is possible.
  - Parallelize formal approval and the proof of eligibilty (they are independent from eachother)

- Information Systems
  - Implement a single database, where
    - Requests are entered directly through an interface
    - E-Mail notifications are sent out to managers when an action has to be taken
    - Travel Requests can be approved directly through an interface
    - Expense reports can be created (e.g. with hotel and transportation information)
    - Receipts/bills can be uploaded instead of handing/archiving them in paper
    - Expense reports can be linked to eachother for easier comparison
    - Expense Reports can be approved directly through an interface
    - An approval of the CFO automatically notifies the finance department
    - Project managers have a view on what their members have spent

As already mentioned, these activities have to be supported by management. The best people should work on implementing the database, senior management should personally invest time to champion and push the project, as well as sufficient resources to train the people that will have to work with the database.



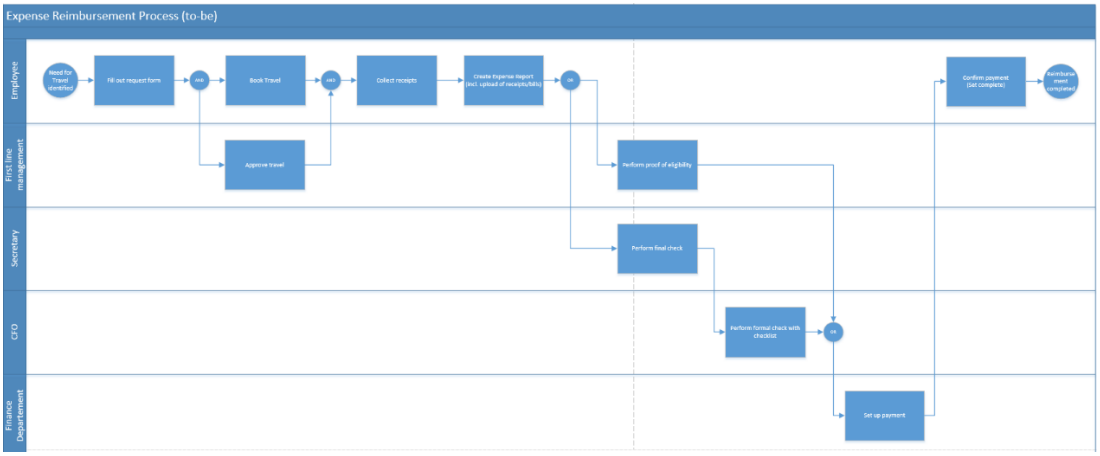
Areas of improvement (current process)

The reengineered process – parallel and database-driven

To make the reengineering project successful, we have made sure to pick a process that has the required breadth. In this case, almost all departments are affected, as most of them have employees traveling regularly. If the process is substantially streamlined, it will result in greater efficiency for most Stark employees, but especially the bottleneck-functions such as the CFO and his/her secretary. It will directly increase employee satisfaction, and indirectly customer satisfaction as well. If management makes sure that the project also has the required depth (by investing a sufficient amount of money to implement the database, educate people on how to use it, etc.), commits fully to it and champions the design proposed by our team, the project will most likely be a success. For the process itself, we have decided to perform all the activities described in the last section:

- Remove the CFOs task of approving travel requests
- Allow the first line manager to deal with expense reports smaller than $500
- Parallelize booking and travel request approval
- Parallelize formal check and proof of eligibilty
- Remove the secretaries tasks of archiving all reports in paper and informing the finance department

This way we have not entirely eliminated waiting times, but reduced them to an amount that will substantially reduce the risk of employees unable to travel, to advance the payment of their business trips and to leave the company, while at the same time increase visibility and responsiveness of all parties involved.



The reengineered process

DATABASE DESIGN

Stark Sealing Solution wants to keep track of employees travel requests and expense reports, compare them to the actual team budgets and associate them with the clients, projects and travel agents involved. Adding this database as the processes' IT backbone enables them to improve their travel request and expense reimbursement process described in the first chapter. This part of the report will focus on (1) listing the entities and attributes used in the database, (2) explain the relationships between the entities, and (3) convert the resulting entity relationship diagram into a set of related tables.

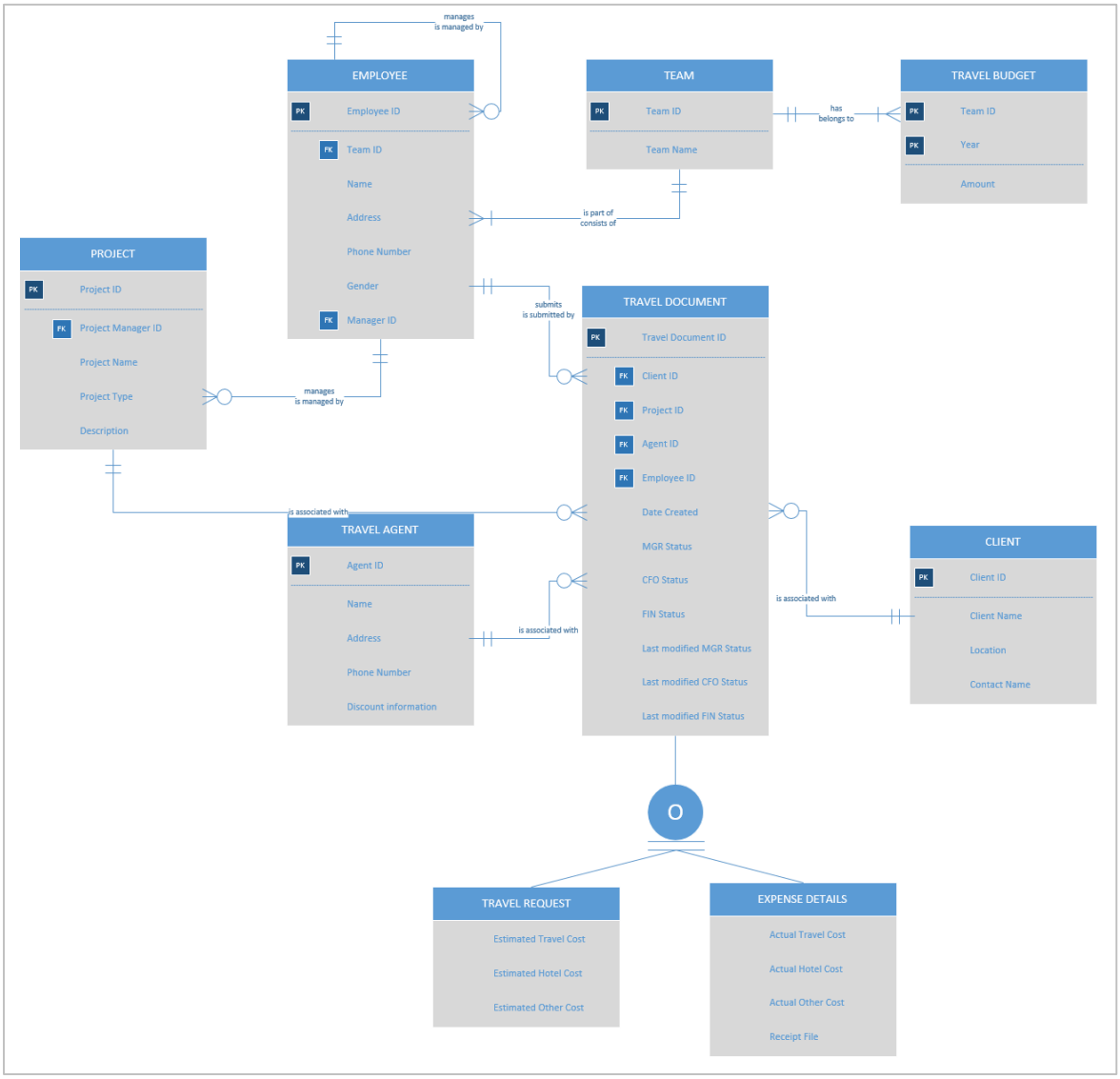
List of entities and attributes

- TEAM – Team ID (PK), Team Name
- TRAVEL BUDGET – Team ID (PK), Year (PK), Amount
- PROJECT – Project ID (PK), Project Manager ID (FK), Project Name, Project Type, Description
- TRAVEL DOCUMENT – Travel Document ID (PK), Client ID (FK), Project ID (FK), Agent ID (FK), Date Created, Manager Status, CFO Status, financial department status, Last modified Manager status, last modified CFO status, last modified financial department status
- TRAVEL REQUEST (Sub) – Estimate Travel Cost, Estimate Hotel Cost, Estimated Other Cost
- EXPENSE DETAILS (Sub) – Actual Travel Cost, Actual Hotel Cost, Actual Other Cost, Receipt File
- CLIENT – Client ID (PK), Client Name, Location, Contact Name
- TRAVEL AGENT – Agent ID (PK), Name, Address, Phone Number, Document Information
- EMPLOYEES – Employee ID (PK), Name, Address, Phone Number, Gender, Manager ID (FK), Team ID (FK)

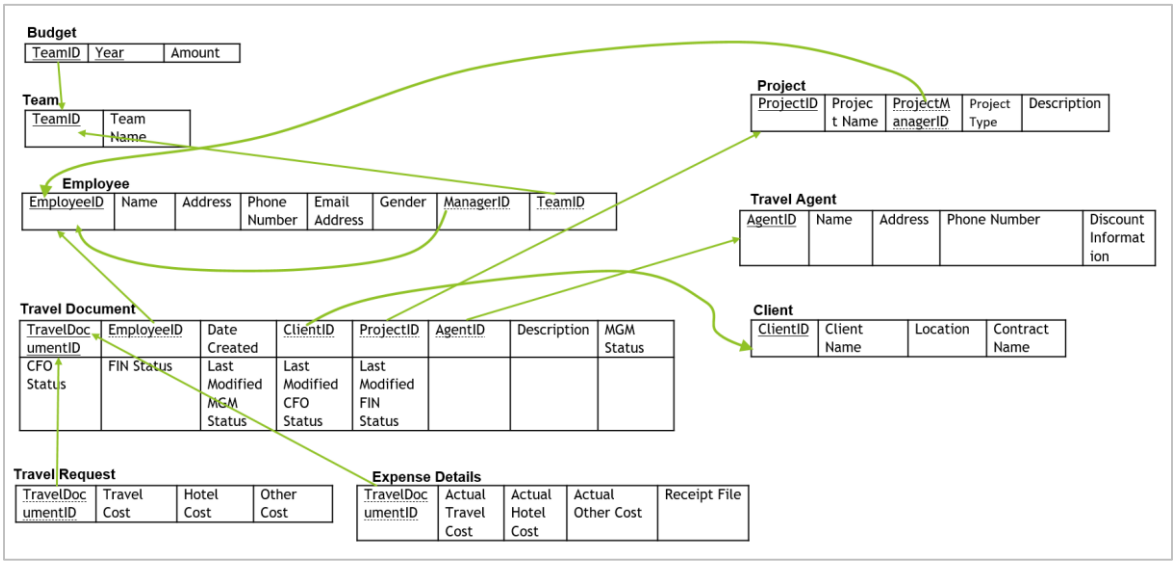
Relationships between entities

- An **EMPLOYEE** is uniquely identified by his employee ID. Additionally, his name, address, phone number, gender and manager are recorded.
- Each **EMPLOYEE** has exactly one manager. A manager is an employee himself and may supervise many employees, but also may not have any employees working directly for him at all.
- An **EMPLOYEE** is part of exactly one **TEAM**, which is uniquely identified by a team ID and has a team name. A team includes at least one and potentially many employees.
- One **TEAM** has one to many **TRAVEL BUDGETS**, as it is updated every year. One budget only applies to one team. The travel budget is uniquely identified by a combination of the team ID and the budgeted year. Of course, the amount should be recorded as well.
- An **EMPLOYEE** may manage zero to many **PROJECTS**, but a single project must be managed by one employee. Projects are uniquely identified by their project ID. The project name, project type and a description are added.
- An **EMPLOYEE** can have submitted many **TRAVEL DOCUMENTS**, but may not have submitted any at all yet. Each travel document can only be submitted by one employee.
- A **TRAVEL DOCUMENT** must be associated with a **TRAVEL AGENT**, **CLIENT** and **PROJECT** (In case no travel agent, client or project can be linked to the travel, an alternative administrative entry will be entered). Each travel agent, client or project may have zero or many travel documents.
- The **TRAVEL DOCUMENT** itself is uniquely identified by a system generated Travel Document ID. As it links employees, projects, travel agents and clients, most primary keys serve as foreign keys in the table. As it combines the two former documents (travel request and expense report), all status information is given in this table as attributes: date created, Manager Status, CFO Status, finance department status, last modified entries for all three statuses.
- A **TRAVEL DOCUMENT** can contain only a travel request up until the point it the expense report is filed. Then, the travel request and the expense details have to exist in parallel. For the travel request, the estimated travel, hotel and other cost are tracked separately. For the expense details, the actual travel, hotel and other cost are tracked separately. A single file containing all receipts must be uploaded with the expense details.

Entity Relationship Diagram



Related Set of Tables



Why we did it this way

1) We combined the travel request entity with the travel expense entity in a supertype/subtype-relationship, because it shares common attributes/format, saves the same purpose, should be easily comparable (for the manager/CFO)
2) Now, the project manager can have a view on all travel cost that occurred on his project
3) The manager and CFO can compare actual and requested cost much easier
4) The manager and the CFO could easily calculate the percentage of their travel budget already used
5) Many views: Open travel requests and expense reports, cost per client, cost per project, cost per employee, cost per team, etc.

DATABASE IMPLEMENTATION

Define Data Types and constraints for each table

To ensure the data inserted into the database correctly, and useful reports can be generated from it, we need to define what the data will look like (numeric, character, etc.). Also, we have to indicate obligatory vs. nullable values (what can be left out), and indicate what primary keys, foreign keys and unique values are.

CLIENT

COLUMN_NAME	DATA_TYPE	NULLABLE	COMMENTS
CLIENT_ID	NUMBER	No	Primary Key
CLIENT_NAME	VARCHAR2(80 BYTE)	No	Unique
LOCATION	VARCHAR2(80 BYTE)	Yes	Null
CONTACT_NAME	VARCHAR2(40 BYTE)	Yes	Null

EMPLOYEE

COLUMN_NAME	DATA_TYPE	NULLABLE	COMMENTS
EMPLOYEE_ID	NUMBER	No	Primary Key
TEAM_ID	NUMBER	No	FK to Team
NAME	VARCHAR2(60 BYTE)	No	Unique (No)
ADDRESS	VARCHAR2(200 BYTE)	Yes	Null 100 characters
PHONE_NUMBER	VARCHAR2(20 BYTE)	Yes	Null
GENDER	VARCHAR2(1 BYTE)	Yes	F – Female M – Male
MANAGER_ID	NUMBER	Yes	FK to Employee

EXPENSE_DETAILS

COLUMN_NAME	DATA_TYPE	NULLABLE	COMMENTS
TRAVEL_DOCUMENT_ID	NUMBER	No	FK to TRAVEL_DOCUMENT & Unique
ACT_TRAVEL_COST	NUMBER(38,0)	Yes	Default 0
ACT_HOTEL_COST	NUMBER(38,0)	Yes	Default 0
ACT_OTHER_COST	NUMBER(38,0)	Yes	Default 0
RECIEPT_FILE_URL	VARCHAR2(200 BYTE)	Yes	URI link to receipt file in S3 server

TRAVEL_REQUEST

COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COMMENTS
TRAVEL_DOCUMENT_ID	NUMBER	No	null	FK to Travel Document
EST_TRAVEL_COST	NUMBER(38,0)	No	0	Default 0
EST_HOTEL_COST	NUMBER(38,0)	No	0	Default 0
EST_OTHER_COST	NUMBER(38,0)	No	0	Default 0

PROJECT

COLUMN_NAME	DATA_TYPE	NULLABLE	COMMENTS
PROJECT_ID	NUMBER	No	Primary Key
PROJECT_MANAGER_ID	NUMBER	No	FK to Employee
PROJECT_NAME	VARCHAR2(80 BYTE)	No	Null
PROJECT_TYPE	VARCHAR2(20 BYTE)	No	Null
DESCRIPTION	VARCHAR2(400 BYTE)	Yes	Null

TEAM

COLUMN_NAME	DATA_TYPE	NULLABLE	COMMENTS
TEAM_ID	NUMBER	No	Primary Key
TEAM_NAME	VARCHAR2(40 BYTE)	No	Unique

TRAVEL AGENT

COLUMN_NAME	DATA_TYPE	NULLABLE	COMMENTS
AGENT_ID	NUMBER	No	Primary Key
ADDRESS	VARCHAR2(80 BYTE)	No	Null
NAME	VARCHAR2(60 BYTE)	No	Unique
PHONE_NUMBER	VARCHAR2(20 BYTE)	No	Null
DISCOUNT_DESCRIPTION	CLOB	Yes	Character Large Object

TRAVEL_BUDGET

COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
TEAMID	NUMBER	No		1	FK to Team
YEAR	VARCHAR2(4 BYTE)	No		2	YYYY format
AMOUNT	NUMBER(38,0)	No	0	3	null

TRAVEL_DOCUMENT

COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COMMENTS
TRAVEL_DOCUMENT_ID	NUMBER	No	Auto generate sequence start from 1 to 99999999	Primary Key
CLIENT_ID	NUMBER	Yes		FK to Client
PROJECT_ID	NUMBER	Yes		FK to Project
AGENT_ID	NUMBER	Yes		FK to Travel Agent
EMPLOYEE_ID	NUMBER	Yes		FK to Employee
CREATED_DATE	TIMESTAMP(6) WITH LOCAL TIME ZONE	Yes	CURRENT_TIMESTAMP	null
MGR_STATUS	NUMBER	No	1	
CFO_STATUS	NUMBER	No	1	
FIN_STATUS	NUMBER	No	1	
MGR_LAST_MODIFICATION	TIMESTAMP(6) WITH LOCAL TIME ZONE	N/A	CURRENT_TIMESTAMP	null
CFO_LAST_MODIFICATION	TIMESTAMP(6) WITH LOCAL TIME ZONE	N/A	CURRENT_TIMESTAMP	null
FIN_LAST_MODIFICATION	TIMESTAMP(6) WITH LOCAL TIME ZONE	N/A	CURRENT_TIMESTAMP	null

MGR STATUS ENUMERATION

Status	Status Description
1	In Process
2	Approved
3	Rejected

CFO_STATUS Enumeration

Status	Status Description
1	In Process
2	Approved
3	Rejected

FIN_STATUS Enumeration

Status	Status Description
1	In Process
2	Approved
3	Rejected

Create database and insert sample data

To create the actual database, we wrote a series of

- drop functions to clean out the database (to be sure no table or sequence from another database or a previous iteration of our database was using the same table names and causing errors)
- sequences to make sure the insertion of new data does not have to include the primary key (ID), because the database automatically generates a new one by counting up the sequence
- create table commands, including the data types and restrictions described in the previous chapter

To test the database, we created tables in excel for each of the relations in the ER diagram, following the rules we set up in the previous chapter. We then imported it into SQL Developer and exported a series of insert commands. All of steps mentioned are covered in the first SQL-document delivered, making it possible to run it on every computer and even multiple times (due to the drop functions included). Examples for each type just mentioned can be found below.

```
DROP TABLE "TRAVEL_REQUEST";
DROP TABLE "EXPENSE_DETAILS";
DROP TABLE "TRAVEL_DOCUMENT";
DROP TABLE "CLIENT";
DROP TABLE "TRAVEL_AGENT";
DROP TABLE "PROJECT";
DROP TABLE "EMPLOYEE";
DROP TABLE "TRAVEL_BUDGET";
DROP TABLE "TEAM";

DROP SEQUENCE "CLIENT_SEQ";
DROP SEQUENCE "EMPLOYEE_SEQ";
DROP SEQUENCE "PROJECT_SEQ";
DROP SEQUENCE "TEAM_SEQ";
DROP SEQUENCE "TRAVEL_AGENT_SEQ";
DROP SEQUENCE "TRAVEL_DOCUMENT_SEQ";
```

```
CREATE TABLE "EMPLOYEE"
 (  "EMPLOYEE_ID" NUMBER,
"TEAM_ID" NUMBER,
"NAME" VARCHAR2(60 BYTE),
"ADDRESS" VARCHAR2(100 BYTE),
"PHONE_NUMBER" VARCHAR2(20 BYTE),
"GENDER" VARCHAR2(1 BYTE),
"MANAGER_ID" NUMBER
 );
```

```
Insert into TEAM (TEAM_ID,TEAM_NAME) values (10117,'Drilling Services');
Insert into TEAM (TEAM_ID,TEAM_NAME) values (10115,'Heating Systems');
Insert into TEAM (TEAM_ID,TEAM_NAME) values (10116,'Transportation Services');
Insert into TEAM (TEAM_ID,TEAM_NAME) values (10114,'Tunnel Services');
```

```
CREATE SEQUENCE  "TRAVEL_DOCUMENT_SEQ"  MINVALUE 1 MAXVALUE 9999999999 INCREMENT BY 1 START WITH 80300 CACHE 20 NOORDER  NOCYCLE ;
```

Trigger

In travel document table, there are three statuses and three modification timestamps which is not effective if user has to update them manually every time user updates status. So we decide to use trigger which is automatic execution procedure on specific event to handle last modification update on each timestamp.

```
CREATE OR REPLACE TRIGGER "TRAVEL_DOCUMENT_UPDATE"
BEFORE UPDATE OF FIN_STATUS,CFO_STATUS,MGR_STATUS ON TRAVEL_DOCUMENT
FOR EACH ROW
BEGIN
  IF :OLD.FIN_STATUS != :NEW.FIN_STATUS THEN
      :NEW.FIN_LAST_MODIFICATION := CURRENT_TIMESTAMP;
  END IF;

  IF :OLD.CFO_STATUS != :NEW.CFO_STATUS THEN
      :NEW.CFO_LAST_MODIFICATION := CURRENT_TIMESTAMP;
  END IF;
```

```
  IF :OLD.MGR_STATUS != :NEW.MGR_STATUS THEN
      :NEW.MGR_LAST_MODIFICATION := CURRENT_TIMESTAMP;
  END IF;
END;
```

Everytime, user make changes to any status, the trigger will be executed and attempt to update last modification timestamp which corresponded to changed status(Automatically).

SQL queries and reports

The final step is come up with queries that extract useful reports for the end user. The following pages describes why the team has chosen the queries provided and show the actual code.

```
----- 1 -----
SHOW A LIST OF EMPLOYEES IN THE DATABASE (MANAGERS CAN SEE WHO HAS ALREADY
TRAVELED OR WHO STILL NEEDS A PROFILE) ------
SELECT EMPLOYEE_ID, NAME, MANAGER_ID, TEAM_ID FROM EMPLOYEE ORDER BY NAME;
```

	EMPLOYEE_ID	NAME	MANAGER_ID	TEAM_ID
1	20024	Adaldrida Baggins	20024	10116
2	20141	Adelbert Fairbairn	20170	10115
3	20103	Adelbert Sackville	20127	10116
4	20004	Alberic Gawkroger	20019	10114
5	20132	Alexander Papst	20161	10114

```
---- 2 ------
LIST ALL TRAVEL REQUESTS WITHOUT MANAGER APPROVAL, INCLUDE PROJECT, TOTAL
REQUESTED AMOUNT AND EMPLOYEE INFORATION (A PROJECT MANAGER CAN SEE PENDING
TRAVEL REQUESTS FOR THEIR PROJECT) -----
SELECT b.TRAVEL_DOCUMENT_ID,
e.PROJECT_NAME,
c.NAME,
f.EST_TRAVEL_COST + f.EST_OTHER_COST + f.EST_HOTEL_COST AS "EST TOTAL AMOUNT"
FROM
TRAVEL_DOCUMENT b
JOIN TRAVEL_REQUEST f ON b.TRAVEL_DOCUMENT_ID = f.TRAVEL_DOCUMENT_ID
JOIN PROJECT e ON b.PROJECT_ID = e.PROJECT_ID
JOIN EMPLOYEE c ON b.EMPLOYEE_ID = c.EMPLOYEE_ID
JOIN TEAM  d ON c.TEAM_ID = d.TEAM_ID
WHERE b.MGR_STATUS < 3;
```

	TRAVEL_DOCUMENT_ID	PROJECT_NAME	NAME	EST TOTAL AMOUNT
1	80171	Viking	Jennifer Faber	1012
2	80172	Moosejaw	Mentha Gardner	1449
3	80173	Bharti Airtel	Melilot Brockhouse	1399
4	80177	Lufthansa	Niklas Konig	1692
5	80178	LA County	Franziska Eisenhower	1668
6	80180	Boston Medical	Ralph Freitag	1911

```
--- 3 ------
IN ADDITION TO THE ONE ABOVE, THIS REPORT SHOWS THE AGE OF EACH REQUEST, WHICH
CAN BE USED TO PRIORITIZE THE APPROVAL PROCESS FOR A MANAGER -----
SELECT b.TRAVEL_DOCUMENT_ID,
e.PROJECT_NAME,
c.NAME,
f.EST_TRAVEL_COST + f.EST_OTHER_COST + f.EST_HOTEL_COST AS "EST TOTAL AMOUNT",
SYSDATE - b.CREATED_DATE AS "days/HH:MM:SS.MSEC"
FROM
TRAVEL_DOCUMENT b
JOIN TRAVEL_REQUEST f ON b.TRAVEL_DOCUMENT_ID = f.TRAVEL_DOCUMENT_ID
JOIN PROJECT e ON b.PROJECT_ID = e.PROJECT_ID
JOIN EMPLOYEE c ON b.EMPLOYEE_ID = c.EMPLOYEE_ID
JOIN TEAM  d ON c.TEAM_ID = d.TEAM_ID
WHERE b.MGR_STATUS < 3
ORDER BY SYSDATE - b.CREATED_DATE DESC;
```

	TRAVEL_DOCUMENT_ID	PROJECT_NAME	NAME	EST TOTAL AMOUNT	days/HH:MM:SS.MSEC
1	80004	LA County	Alberic Gawkroger	1186	+01 01:15:57.471000
2	80049	FIDM	Menegilda Chubb	520	+01 01:15:57.471000
3	80001	Moosejaw	Arnor Banks	1245	+01 01:15:57.471000
4	80000	Viking	Ralf Busch	964	+01 01:15:57.471000
5	80045	Moosejaw	Eglantine Goodbody	1552	+01 01:15:57.471000
6	80041	FIDM	Laura Cole	1760	+01 01:15:57.471000

```
----- 4 ------
COMPARE TRAVEL REQUEST AND EXPENSE DETAILS ON EACH TRAVEL DOCUMENT BY EXPENSE
CATEGORIES (TRAVEL, HOTEL, OTHER) AND ALSO SHOW RECEIPT_FILE IF AVAILABLE
SELECT C.EMPLOYEE_ID, C.NAME AS EMPLOYEE_NAME ,
A.TRAVEL_DOCUMENT_ID, A.CLIENT_ID, A.PROJECT_ID, A.AGENT_ID, trunc(A.CRE-
ATED_DATE) AS "Created DATE",
B.EST_TRAVEL_COST || '/' || NVL2(D.ACT_TRAVEL_COST,
TO_CHAR(D.ACT_TRAVEL_COST),'N/A') AS "TRAVEL_COST(EST/ACT)",
B.EST_HOTEL_COST || '/' || NVL2(D.ACT_HOTEL_COST, TO_CHAR(D.ACT_HO-
TEL_COST),'N/A') AS "HOTEL_COST(EST/ACT)",
B.EST_TRAVEL_COST || '/' || NVL2(D.ACT_OTHER_COST,
TO_CHAR(D.ACT_OTHER_COST),'N/A') AS "OTHER_COST(EST/ACT)",
NVL(D.RECIEPT_FILE_URL, ' RECEIPT NOT AVAILABLE'),
A.MGR_STATUS AS MANAGER_APPROVAL,
A.CFO_STATUS AS CFO_APPROVAL,
A.FIN_STATUS AS FINANCIAL_APPROVAL
FROM TRAVEL_DOCUMENT A JOIN TRAVEL_REQUEST B ON A.TRAVEL_DOCUMENT_ID =
B.TRAVEL_DOCUMENT_ID
JOIN EXPENSE_DETAILS D ON A.TRAVEL_DOCUMENT_ID = D.TRAVEL_DOCUMENT_ID
JOIN EMPLOYEE C ON A.EMPLOYEE_ID = C.EMPLOYEE_ID;
```

	EM...	EMPLOYEE_NAME	TRA...	CLIE...	PRO...	AGE...	Created DATE	TRAVEL_...	HOTEL_...	OTHER_...	NVL(D.RECIEPT_FILE_URL,'...	.	...	..
1	20000	Ralf Busch	80000	50000	30000	40000	06-DEC-15	281/940	547/741	281/538	RECEIPT NOT AVAILABLE	2	1	1
2	20001	Arnor Banks	80001	50001	30001	40001	06-DEC-15	686/973	478/586	686/627	RECEIPT NOT AVAILABLE	1	1	1
3	20002	Haiduc Lothran	80002	50002	30002	40002	06-DEC-15	859/513	476/168	859/656	RECEIPT NOT AVAILABLE	1	1	1
4	20003	Asphodel Hayward	80003	50003	30003	40003	06-DEC-15	653/979	148/147	653/733	RECEIPT NOT AVAILABLE	3	3	3
5	20004	Alberic Gawkroger	80004	50004	30004	40000	06-DEC-15	361/610	676/816	361/169	RECEIPT NOT AVAILABLE	2	1	1

```
--- 5 -----
HOW MANY TRAVEL DOCUMENTS HAVE BEEN CREATED BY EACH EMPLOYEE, GROUPED BY
EMPLOYEE NAME AND PROJECT (CHECK FOR FRAUDULENT ACTIVITY) ----
SELECT DISTINCT b.NAME, c.PROJECT_NAME, count(1) AS TOTAL_DOCUMENT_CREATED
FROM TRAVEL_DOCUMENT a
JOIN EMPLOYEE b
ON a.EMPLOYEE_ID = b.EMPLOYEE_ID
JOIN PROJECT c
ON a.PROJECT_ID = c. PROJECT_ID
WHERE a.MGR_STATUS != 3
GROUP BY b.NAME, c.PROJECT_NAME;
```

	NAME	PROJECT_NAME	TOTAL_DOCUMENT_CREATED
1	Ralf Busch	Viking	1
2	Ralph Jung	Viking	1
3	Pimpernel Took-Brandybuck	Viking	1
4	Paul Keller	Viking	1
5	Daniel Schneider	Viking	1

```
--- 6 -----
SEE HOW MANY REPORTS HAVE WHICH STATUS (STATISTICAL INFORMATION TO SEE IF
PROCESS AND DATABASE ARE PERFORMING WELL) ----
SELECT CASE MGR_STATUS
WHEN 1 THEN 'In Process'
WHEN 2 THEN 'Approved'
```

```
WHEN 3 THEN 'Rejected'
END AS "MANAGER_STATUS",
count(1)
FROM TRAVEL_DOCUMENT
GROUP BY MGR_STATUS;
```

	MANAGER_STATUS	COUNT(1)
1	In Process	25
2	Approved	49
3	Rejected	126

```
--- 7 -----
TEAM BUDGET AND TRAVEL SPENT BY TEAM, FOR INSTANCE IN YEAR 2015, SHOW
PERCENTAGE OF TRAVEL BUDGET USED COMPARED TO THE ONE AVAILABLE (CHECK IF TEAM
MAY GO OVER BUDGET) -----

SELECT AA.TEAM_ID, AA.TEAM_NAME, AA.TOTAL_SPENT, BB.YEAR, BB.AMOUNT,
(AA.TOTAL_SPENT/BB.AMOUNT) * 100 AS PCT_USED
FROM
(
  SELECT D.TEAM_ID, D.TEAM_NAME, SUM(A.TOTAL_SPENT) AS "TOTAL_SPENT"
  FROM
  (
    SELECT TRAVEL_DOCUMENT_ID,
    ACT_HOTEL_COST + ACT_OTHER_COST + ACT_TRAVEL_COST AS "TOTAL_SPENT" FROM
EXPENSE_DETAILS
  ) A
  JOIN TRAVEL_DOCUMENT B ON
  A.TRAVEL_DOCUMENT_ID = B.TRAVEL_DOCUMENT_ID
  JOIN EMPLOYEE C ON
  B.EMPLOYEE_ID = C.EMPLOYEE_ID
  JOIN TEAM D ON
  C.TEAM_ID = D.TEAM_ID
  WHERE
  EXTRACT(year FROM B.CREATED_DATE) = &year
  GROUP BY D.TEAM_ID,D.TEAM_NAME
) AA
JOIN
TRAVEL_BUDGET BB
ON AA.TEAM_ID = BB.TEAMID AND BB.YEAR = &budget_year;
```

	TEAM_ID	TEAM_NAME	TOTAL_SPENT	YEAR	AMOUNT	PCT_USED
1	10115	Heating Systems	49795	2015	100000	49.795
2	10114	Tunnel Services	62108	2015	200000	31.054
3	10116	Transportation Services	27813	2015	50000	55.626
4	10117	Drilling Services	15591	2015	100000	15.591

```
---- 8 -----
TRAVEL EXPENSES EACH PROJECT HAS CAUSED SO FAR, ORDERED BY AMOUNT DESCENDING
(CHECK IF PROJECT WILL EXCEED TRAVEL BUDGET) ---
SELECT
e.PROJECT_NAME,
SUM(f.ACT_TRAVEL_COST + f.ACT_OTHER_COST + f.ACT_HOTEL_COST)||' €' AS "TRAVEL
EXPENSES TO DATE"
FROM
TRAVEL_DOCUMENT b
JOIN EXPENSE_DETAILS f ON b.TRAVEL_DOCUMENT_ID = f.TRAVEL_DOCUMENT_ID
```

```
JOIN PROJECT e ON b.PROJECT_ID = e.PROJECT_ID
JOIN EMPLOYEE c ON b.EMPLOYEE_ID = c.EMPLOYEE_ID
JOIN TEAM  d ON c.TEAM_ID = d.TEAM_ID
GROUP BY e.PROJECT_NAME
HAVING SUM((f.ACT_TRAVEL_COST + f.ACT_OTHER_COST + f.ACT_HOTEL_COST))>2000
ORDER BY SUM((f.ACT_TRAVEL_COST + f.ACT_OTHER_COST + f.ACT_HOTEL_COST)) DESC;
```

	PROJECT_NAME	TRAVEL EXPENSES TO DATE
1	Lufthansa	25731 €
2	LA County	25078 €
3	FIDM	21231 €
4	Bharti Airtel	19235 €
5	Viking	18600 €
6	Moosejaw	16225 €
7	Vital Alarm	13831 €
8	Boston Medical	12787 €

```
---- 9 -----
AMOUNT THE COMPANY HAS SPENT TO DATE (STATISTICAL INFORMATION FOR CFO TO PLAN
FOR NEXT PERIOD) ----
SELECT
SUM(f.ACT_TRAVEL_COST + f.ACT_OTHER_COST + f.ACT_HOTEL_COST)||' ' AS "TRAVEL
EXPENSES TO DATE"
FROM
TRAVEL_DOCUMENT b
JOIN EXPENSE_DETAILS f ON b.TRAVEL_DOCUMENT_ID = f.TRAVEL_DOCUMENT_ID
JOIN PROJECT e ON b.PROJECT_ID = e.PROJECT_ID
JOIN EMPLOYEE c ON b.EMPLOYEE_ID = c.EMPLOYEE_ID
JOIN TEAM  d ON c.TEAM_ID = d.TEAM_ID;
```

	TRAVEL EXPENSES TO DATE
1	155307 €

```
---- 10 ----
LAURA MUGWORT HAS MARRIED, RENAME HER TO LAURA WURSTER ----
UPDATE employee SET name = 'Laura Wurster' WHERE employee_id = 20106;


---- 11 ----
GET THE TOTAL OF PROJECTS PER EMPLOYEE, THE TOTAL AND AVERAGE AMOUNT SPENT PER
PROJECT (DISCOVER FRAUDULENT ACTIVITIES) -----
SELECT  AA.name,AA.team_name ,
AA.ACT_HOTEL_COST+AA.ACT_TRAVEL_COST+ AA.ACT_OTHER_COST as "the amount he
spent",
AB."COUNT_PROJect" ,
(AA.ACT_HOTEL_COST+AA.ACT_TRAVEL_COST+ AA.ACT_OTHER_COST)/AB."COUNT_PROJect"
as "amount/number of projects"
FROM
(select e.name , t.team_name , e.EMPLOYEE_ID,
d.ACT_HOTEL_COST,d.ACT_TRAVEL_COST, d.ACT_OTHER_COST
from EMPLOYEE e join team t
on e.TEAM_ID = t.TEAM_ID
join TRAVEL_DOCUMENT r
on e.EMPLOYEE_ID = r.EMPLOYEE_ID
join EXPENSE_DETAILS d
```

```
on r.TRAVEL_DOCUMENT_ID = d.TRAVEL_DOCUMENT_ID) AA
join
(select count(p.project_id) AS "COUNT_PROJect" , e.name
from project p join  TRAVEL_DOCUMENT r
on r.PROJECT_ID = p.PROJECT_ID
join EMPLOYEE e
on e.EMPLOYEE_ID = r.EMPLOYEE_ID
group by e.name)AB
ON AA.name = AB.name;
```

	NAME	TEAM_NAME	the amount he spent	COUNT_PROJect	amount/number of projects
1	Estella Gardner	Tunnel Services	1091	1	1091
2	Marmaduc Mugwort	Tunnel Services	1394	1	1394
3	Eglantine Goodbody	Tunnel Services	2350	1	2350
4	Diana Klein	Tunnel Services	828	1	828
5	Pamphila Diggle	Transportation Services	2249	1	2249
6	Lavinia Roper	Heating Systems	1284	1	1284

```
---- 12 ------
NUMBER OF TRAVEL REQUESTS PER TEAM (STATISTICAL INFORMATION FOR CFO SEE WHICH
TEAMS TRAVEL MOST AND MAY NEED HIGHER TRAVEL BUDGETS) -----
SELECT c.team_name, count(1) AS "Number Of Travel Request"
from travel_request a join travel_document b
on a.travel_document_id= b.travel_document_id
join employee c
on b.employee_id= c.employee_id
join TEAM d
on b.team_id = d.TEAMID
group by d.team_name;
```

	TEAM_NAME	Number Of Travel Request
1	Heating Systems	63
2	Transportation Services	36
3	Tunnel Services	84
4	Drilling Services	17

```
--- 13 -----
ADDING, RENAMING AND DROPPING A CONSTRAINT THAT LIMITS TEAM BUDGETS ----
ALTER TABLE TRAVEL_BUDGET
ADD CONSTRAINT must_less_than_billion CHECK (AMOUNT < 100000000);

ALTER TABLE TRAVEL_BUDGET
RENAME CONSTRAINT must_less_than_billion TO must_less_than_billion_Yeah;


ALTER TABLE TRAVEL_BUDGET
DROP CONSTRAINT must_less_than_billion_Yeah;


-- 14 -----
For each manager, give out a list of open travel requests and expense reports,
compare travel requests and expense (to make approval process focus on
exceeded budgets (give manager an overview which expense reports can be
approved quickly, and which exceeded the allowed budget and have to be looked
at more closely) --- ---
```

```
SELECT A.TRAVEL_DOCUMENT_ID, B.NAME, B.GENDER,
CASE A.MGR_STATUS
WHEN 1 THEN 'In Process'
WHEN 2 THEN 'Approved'
WHEN 3 THEN 'Rejected'
END AS STATUS,
TR.EST_HOTEL_COST + TR.EST_OTHER_COST + TR.EST_TRAVEL_COST AS
"ESTIMATED_COST",
NVL2(ED.TRAVEL_DOCUMENT_ID, TO_CHAR(ED.ACT_HOTEL_COST + ED.ACT_TRAVEL_COST +
ED.ACT_OTHER_COST), 'N/A') AS "ACTUAL COST",
NVL2(ED.TRAVEL_DOCUMENT_ID, TO_CHAR((TR.EST_HOTEL_COST + TR.EST_OTHER_COST +
TR.EST_TRAVEL_COST) - (ED.ACT_HOTEL_COST + ED.ACT_TRAVEL_COST +
ED.ACT_OTHER_COST)), 'N/A') AS "DIFF(EST - ACT)",
A.CREATED_DATE  FROM
TRAVEL_DOCUMENT A JOIN EMPLOYEE B
ON A.EMPLOYEE_ID = B.EMPLOYEE_ID
JOIN TRAVEL_REQUEST TR ON A.TRAVEL_DOCUMENT_ID = TR.TRAVEL_DOCUMENT_ID
LEFT JOIN EXPENSE_DETAILS ED ON A.TRAVEL_DOCUMENT_ID = ED.TRAVEL_DOCUMENT_ID
JOIN EMPLOYEE C
ON B.MANAGER_ID = C.EMPLOYEE_ID
WHERE MGR_STATUS = 2
AND C.EMPLOYEE_ID = &MANAGER_ID;
```

	TRAVEL_DOCUMENT_ID	NAME	GENDER	STATUS	ESTIMATED_COST	ACTUAL COST	DIFF(EST - ACT)	CREATED_DATE
1	80032	Alexander Sommer	M	In Process	2179	1374	805	06-DEC-15 04.50.22.529000000 PM
2	80033	Ralph Jung	M	In Process	1795	603	1192	06-DEC-15 04.50.22.529000000 PM
3	80037	Gundobad Gardner	M	In Process	1705	1354	351	06-DEC-15 04.50.22.529000000 PM
4	80038	Miranda Noakes	F	In Process	292	1568	-1276	06-DEC-15 04.50.22.529000000 PM

```
--- 15 ----
COMPARE ALL EXPENSES TO ESTIMATED TRAVEL COSTS ON ALL TRAVEL DOCUMENTS NO
MATTER WHO IS THE MANAGER (IF CFO WANTS TO SEE HOW OFTEN EMPLOYEES EXCEED
BUDGET) ----
SELECT A.TRAVEL_DOCUMENT_ID, A.EST_HOTEL_COST + A.EST_OTHER_COST +
A.EST_TRAVEL_COST AS "ESTIMATED_COST",
NVL2(B.TRAVEL_DOCUMENT_ID, TO_CHAR(B.ACT_HOTEL_COST + B.ACT_OTHER_COST +
B.ACT_TRAVEL_COST), 'N/A') AS "ACTUAL_COST",
NVL2(B.TRAVEL_DOCUMENT_ID, TO_CHAR((A.EST_HOTEL_COST + A.EST_OTHER_COST +
A.EST_TRAVEL_COST) - (B.ACT_HOTEL_COST + B.ACT_OTHER_COST +
B.ACT_TRAVEL_COST)), 'N/A') AS "DIFF"
FROM TRAVEL_REQUEST A
LEFT JOIN
EXPENSE_DETAILS B
ON A.TRAVEL_DOCUMENT_ID = B.TRAVEL_DOCUMENT_ID;
```

	TRAVEL_DOCUMENT_ID	Estimated Cost	Actual Cost	Different
1	80000	964 €	2219 €	-1255 €
2	80001	1245 €	2186 €	-941 €
3	80003	1171 €	1859 €	-688 €
4	80004	1186 €	1595 €	-409 €
5	80005	2156 €	2232 €	-76 €
6	80006	742 €	1343 €	-601 €

```
---- 16 ----
REPORTS THAT HAVE BEEN ISSUED WITHIN THE LAST 24 HOURS (COULD BE USED AS AN
AUTOMATED DAILY REPORT TO GENERATE STATISTICAL VIEW FOR CFO ON HOW MANY
REPORTS HAVE BEEN CREATED, ARE IN PROCESS OR APPROVED ----
CREATE OR REPLACE FORCE VIEW "GET_LESS_THAN_DAY_TD" ("TRAVEL_DOCUMENT_ID",
"CLIENT_ID", "PROJECT_ID", "AGENT_ID", "EMPLOYEE_ID", "DIFF", "MANAGER_STA-
TUS", "CFO_STATUS", "FINANCIAL_STATUS")
AS
  SELECT "TRAVEL_DOCUMENT_ID","CLIENT_ID","PROJECT_ID","AGENT_ID","EM-
PLOYEE_ID","DIFF","MANAGER_STATUS","CFO_STATUS","FINANCIAL_STATUS"
  FROM
  (SELECT TRAVEL_DOCUMENT_ID, CLIENT_ID, PROJECT_ID, AGENT_ID, EMPLOYEE_ID,
  SYSTIMESTAMP - CREATED_DATE AS DIFF,
  CASE MGR_STATUS
    WHEN 1 THEN 'In Process'
    WHEN 2 THEN 'Approved'
    WHEN 3 THEN 'Rejected'
  END AS "MANAGER_STATUS",
  CASE CFO_STATUS
    WHEN 1 THEN 'In Process'
    WHEN 2 THEN 'Approved'
    WHEN 3 THEN 'Rejected'
  END AS "CFO_STATUS",
  CASE FIN_STATUS
     WHEN 1 THEN 'In Process'
    WHEN 2 THEN 'Approved'
    WHEN 3 THEN 'Rejected'
  END AS "FINANCIAL_STATUS"
  FROM TRAVEL_DOCUMENT)
 WHERE EXTRACT(day from DIFF) < 1;

--- 17 ----
INDEX TO IMPROVE PERFORMANCE ON EMPLOYEE_ID IN TRAVEL DOCUMENT TABLE ----
CREATE INDEX index_employee ON TRAVEL_DOCUMENT(employee_id);

--- 18 ----
UNION: GET THE FIRST FIVE MEN AND THE FIRST FIVE WOMEN FROM EMPLOYEES ----
SELECT *
FROM
(
SELECT * FROM
(
SELECT *
FROM EMPLOYEE
WHERE GENDER = 'M'
ORDER BY NAME
)
WHERE ROWNUM <= 5
UNION ALL
SELECT * FROM
(
SELECT *
FROM EMPLOYEE
WHERE GENDER = 'F'
ORDER BY NAME
)
WHERE ROWNUM <= 5
);
```

	EMPLOYEE_ID	TEAM_ID	NAME	ADDRESS	PHONE_NUMBER	GENDER	MANAGER_ID
1	20141	10115	Adelbert Fairbairn	BleibtreustraÃŸe 7 - 79790 KÃ¼ssaberg	07741 64 18 60	M	20170
2	20103	10116	Adelbert Sackville	Storkower Strasse 72 - 56235 Ransbach-Baumbach	02623 98 24 12	M	20127
3	20004	10114	Alberic Gawkroger	Schaarsteinweg 51 - 93346 Ihrlerstein	09441 77 40 46	M	20019
4	20132	10114	Alexander Papst	Koepenicker Str. 94 - 56357 Oelsberg	06772 22 59 13	M	20161
5	20032	10115	Alexander Sommer	Grosse Praesidenten Str. 13 - 74855 HaÃŸmersheim	06261 16 31 40	M	20024
6	20024	10116	Adaldrida Baggins	HeiligengeistbrÃ¼cke 43 - 91533 Rothenburg	09861 62 97 08	F	20024
7	20190	10114	Amethyst Boffin	Augsburger Strasse 6 - 53520 Rodder	02693 89 66 29	F	20199
8	20078	10114	Anke Boehm	Amsinckstrasse 61 - 7623 Hermsdorf	035057 98 66	F	20094
9	20058	10115	Anne Bohm	Ollenhauer Str. 19 - 70599 Stuttgart Hohenheim	0711 19 83 56	F	20071
10	20188	10114	Antje Maurer	Los-Angeles-Platz 87 - 22459 Hamburg Schnelsen	040 43 59 83	F	20199

```
--- 19 ----
INTERSECT: GET THE TOTAL NUMBER OF TRAVEL DOCUMENTS FOR ALL PROJECTS WHERE
THERE MORE THAN 15 and less than 30 ----
SELECT *
FROM
(SELECT A.PROJECT_NAME , COUNT(1) AS TOTAL_DOCUMENT FROM
PROJECT A JOIN TRAVEL_DOCUMENT B
ON A.PROJECT_ID = B.PROJECT_ID
GROUP BY A.PROJECT_NAME
HAVING COUNT(1) > 15)
INTERSECT
(SELECT A.PROJECT_NAME , COUNT(1) AS TOTAL_DOCUMENT FROM
PROJECT A JOIN TRAVEL_DOCUMENT B
ON A.PROJECT_ID = B.PROJECT_ID
GROUP BY A.PROJECT_NAME
HAVING COUNT(1) < 30);
```

	PROJECT_NAME	TOTAL_DOCUMENT
1	Bharti Airtel	21
2	Boston Medical	21
3	FIDM	23
4	Moosejaw	23
5	Viking	23
6	Vital Alarm	21