Sales & Trading

PWM

Investment Banking

AlanCorp.

"Providing financial services for institutional, corporate & personal clients since 1975"

A Database Design Proposal

Dietrich Mosel

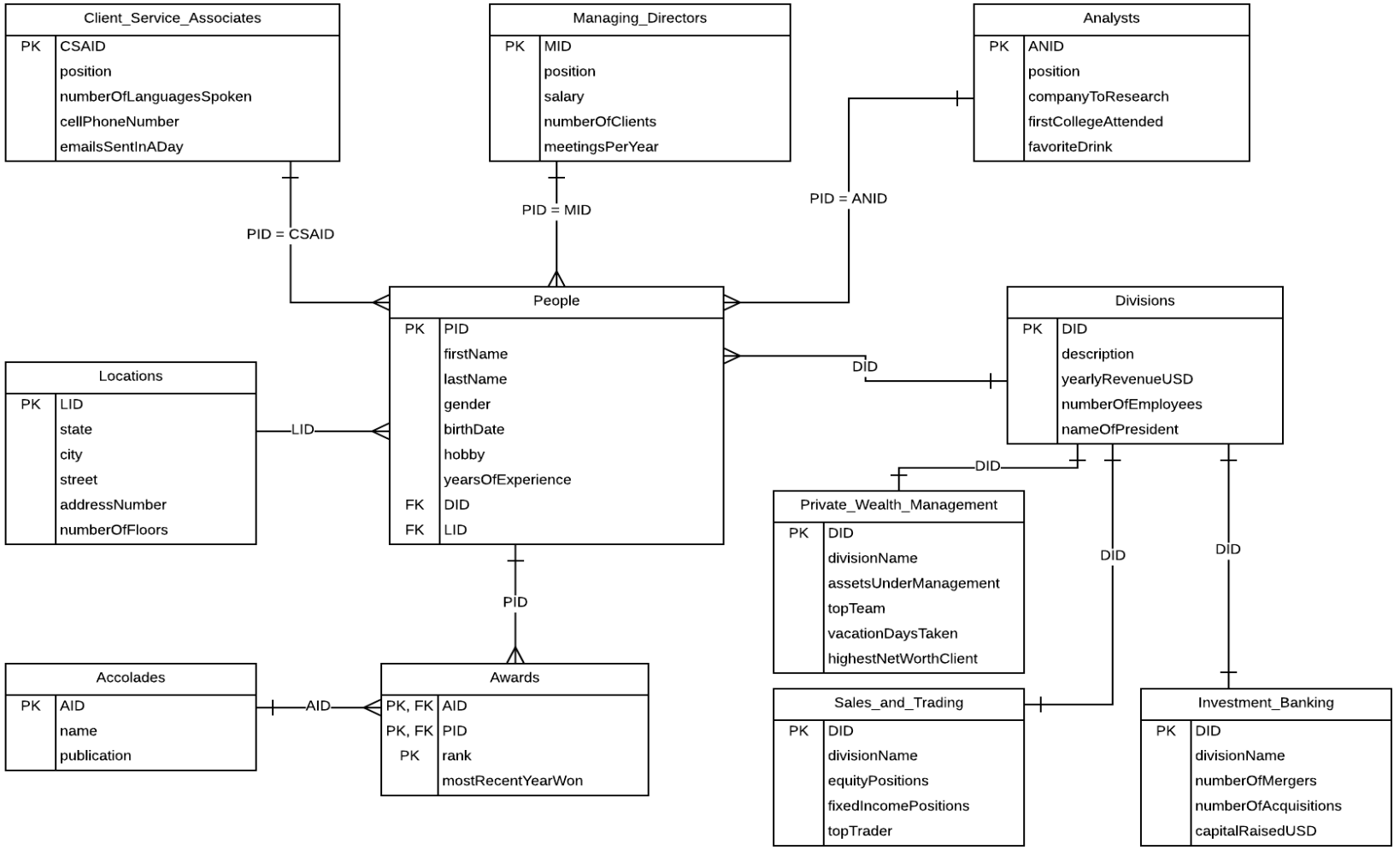
December 7th, 2017

Table of Contents

Executive Summary

AlanCorp. was formed in 1975 by Alan Labouseur himself, to better provide select financial services to clients across the United States. Tired of being treated second-hand, Mr. Labouseur sought to provide exceptional customer services for clients like himself that felt underappreciated. You can see from the following tables in this relational database that the company is highly profitable across its three key divisions. These are comprised of Private Wealth Management, Investment Banking and Sales and Trading. The first seeks to manage stores of wealth for high and ultra high net worth clients while the Investment Banking segment pounces on opportunities to help businesses in the complex financial implementations of mergers and acquisitions. Finally, the Sales and Trading division is responsible for buying and selling equities and fixed income securities for institutional and outside clients.

Entity-Relationship Diagram



Create Table Statements

People table

Employees of AlanCorp. can work in one of three divisions which entail Private Wealth Management, Sales and Trading as well as Investment Banking. This table is utilized to separate basic personnel information.

```
DROP TABLE People CASCADE;

CREATE TABLE IF NOT EXISTS People (
        PID int NOT NULL UNIQUE,
        firstName varchar(255) NOT NULL,
        lastName varchar(255) NOT NULL,
        gender varchar(255) NOT NULL,
        birthDate date NOT NULL,
        hobby varchar(255),
        yearsOfExperience int NOT NULL,
        DID int NOT NULL,
        LID int,
        CONSTRAINT gender_type CHECK (gender = 'F' OR gender = 'M'),
        PRIMARY KEY (PID),
        FOREIGN KEY (DID) REFERENCES Divisions (DID),
        FOREIGN KEY (LID) REFERENCES Locations (LID)
);
```

Sample Data

pid integer	firstname character varying(255)	lastname character varying(255)	gender character varying(255)	birthdate date	hobby character varying(255)	yearsofexperience integer	did integer	lid integer
1	Jesse	Korris	M	1960-05-17	Skiing	27	2	2
2	Mitch	Krout	M	1970-02-06	Hangliding	17	3	1
3	Greg	King	M	1943-07-25	Golfing	5	1	3
4	Jessica	Albert	F	1966-03-03	Billiards	2	1	3
5	Angela	Paulson	F	1972-08-01	Soccer	10	3	1
6	Dietrich	Mosel	M	1996-07-10	Running	0	1	3
7	Alan	Labouseur	M	1965-11-18	Portal	14	2	2
8	Sam	Chantly	F	1987-01-22	Cooking	6	2	2
9	Chris	Badolato	M	1957-12-12	Restoring Cars	32	1	3
10	Emily	Roberts	F	1992-09-19	Traveling	3	3	1
11	Gianna	Louro	F	1957-03-14		26	3	1
12	Mark	Valentino	M	1963-07-21	Card Counting	14	1	3
13	Vinny	Donatacci	M	1987-03-07	Volunteering	7	2	2
14	Kenny	Walshek	M	1969-05-16	Teaching	15	1	3
15	Michaela	Murray	F	1959-04-01	Wine Tasting	8	3	1
16	Bianca	Luparello	F	1974-06-03	Clubbing	2	2	2
17	John	Lee	M	1957-10-10	Video Games	36	1	3
18	Kyle	Hannafin	M	1977-09-20	Basketball	22	1	3
19	Olivia	Cray	F	1984-07-27	Coding	18	2	2
20	Tom	Slattery	M	1988-12-29	Snow Boarding	13	3	1

Functional dependencies

PID → firstname, lastname, gender, birthDate, hobby, yearsOfExperience, DID, LID

Client_Service_Associates table

This is one of the specific job functions an employee can engage in. A CSA is a clients first point of contact, servicing needs of both consumers and the team they work for. It is an entry level position.

```
DROP TABLE Client_Service_Associates CASCADE;

CREATE TABLE IF NOT EXISTS Client_Service_Associates (
        CSAID int NOT NULL UNIQUE,
        position varchar(255) NOT NULL DEFAULT 'Client Service Associate',
        numberOfLanguagesSpoken int NOT NULL,
        cellPhoneNumber int NOT NULL,
        emailsSentInADay int NOT NULL,
        PRIMARY KEY (CSAID),
        FOREIGN KEY (CSAID) REFERENCES people(PID)
);
```

Functional dependencies

CSAID → position,

numberOfLanguagesSpoken,

cellPhoneNumber, emailsSentInADay

Sample Data

csaid integer	position character varying(255)	numberoflanguagesspoken integer	cellphonenumber integer	emailssentinaday integer
3	Client Service Associate	3	51631856	30
4	Client Service Associate	2	64951324	50
6	Client Service Associate	4	65385109	34
8	Client Service Associate	1	54675638	45
10	Client Service Associate	1	93756475	25
15	Client Service Associate	2	66254091	66
16	Client Service Associate	3	29061534	89

Managing_Directors table

Another job function employees of AlanCorp. can take advantage of. However, this specific job requires years and years of experience with a proven track record of success, as the managing directors handle most of the larger decisions.

```
DROP TABLE Managing_Directors CASCADE;

CREATE TABLE IF NOT EXISTS Managing_Directors (
        MID int NOT NULL UNIQUE,
        position varchar(255) NOT NULL DEFAULT 'Managing Director',
        salary int NOT NULL,
        numberOfClients int NOT NULL,
        meetingsPerYear int NOT NULL,
        PRIMARY KEY (MID),
        FOREIGN KEY (MID) REFERENCES people(PID)
);
```

Sample Data

mid integer	position character varying(255)	salary integer	numberofclients integer	meetingsperyear integer
1	Managing Director	250778	275	223
9	Managing Director	176348	236	187
11	Managing Director	326331	401	356
17	Managing Director	108450	210	142
19	Managing Director	223791	260	218
20	Managing Director	223791	260	218

Functional dependencies

MID → position, salary, numberOfClients, meetingsPerYear

Analysts table

Analysts are the last job responsibility employees of AlanCorp. can hold. The analysts are given the task of researching companies for the firm to invest in or short sell for clients across the board. This position is used as a springboard to get the managing director job.

```
DROP TABLE Analysts CASCADE;

CREATE TABLE IF NOT EXISTS Analysts (
        ANID int NOT NULL UNIQUE,
        position varchar(255) NOT NULL DEFAULT 'Analyst',
        companyToResearch varchar(255) NOT NULL,
        firstCollegeAttended varchar(255) NOT NULL,
        favoriteDrink varchar(255),
        PRIMARY KEY (ANID),
        FOREIGN KEY (ANID) REFERENCES people(PID)
);
```

Sample Data

anid integer	position character varying(255)	companytoresearch character varying(255)	firstcollegeattended character varying(255)	favoritedrink character varying(255)
2	Analyst	Apple	Harvard	Dirty Martini
5	Analyst	Boeing	Marist	Tokyo Mule
7	Analyst	ImmunoGen	UCLA	Four Score
12	Analyst	Skyworks Solutions	UPenn	Bloodhound
13	Analyst	MongoDB	Princeton	Grog
14	Analyst	Nvidia	Stony Brook	Sake Bomb
18	Analyst	Google	Lehigh	Long Island Iced Tea

Functional dependencies

ANID → position, companyToResearch,

firstCollegeAttended, favoriteDrink

Divisions table

This table highlights the three divisions of AlanCorp. that are responsible for driving the firm's revenues. These entail, Private Wealth Management, Sales and Trading and Investment Banking. Each has different functions within the company and employs varying numbers of people who are managed by a president.

```
DROP TABLE Divisions CASCADE;

CREATE TABLE IF NOT EXISTS Divisions (
        DID int NOT NULL UNIQUE,
        description varchar(255) NOT NULL,
        yearlyRevenueUSD int NOT NULL,
        numberOfEmployees int NOT NULL,
        nameofPresident varchar(255) NOT NULL,
        PRIMARY KEY (DID)
);
```

Functional dependencies

DID → description, yearlyRevenueUSD, numberOfEmployees, nameOfPresident

Sample Data

did integer	description character varying(255)	yearlyrevenueusd integer	numberofemployees integer	nameofpresident character varying(255)
1	Private Wealth Management	500000000	1500	John McCmullen
2	Sales and Trading	750000000	2000	Craig Keystone
3	Investment Banking	900000000	2500	Todd Ferguson

Investment_Banking table

The Investment Banking division concentrates on helping companies deal with financial stresses related to mergers or acquisitions with other firms. AlanCorp. makes money off fees charged for this assistance being provided. The capital raised by these mergers and acquisitions is evident within the table as well.

Sample Data

did integer	divisionname character varying(255)	numberofmergers integer	numberofacquisitions integer	capitalraised integer
3	Investment Banking	27	56	1320564338

```
DROP TABLE Investment_Banking CASCADE;

CREATE TABLE IF NOT EXISTS Investment_Banking (
        DID int NOT NULL UNIQUE,
        Divisionname varchar(255) NOT NULL,
        numberOfMergers int NOT NULL,
        numberOfAcquisitions int NOT NULL,
        capitalRaised int NOT NULL,
        PRIMARY KEY (DID)
);
```

Functional dependencies

DID → divisionName, numberOfMergers, numberofAcquisitions, capitalRaised

Sales_and_Trading table

Sales and Trading is the second division of AlanCorp. where traders buy and sell either equity or fixed income securities in an effort to gain profit for institutional and outside clients. This table exhibits the amount of equity and fixed income positions along with the trader who has made the most money in the given fiscal year.

Sample Data

did integer	divisionname character varying(255)	equitypositions integer	fixedincomepositions integer	toptrader character varying(255)
2	Sales and Trading	10260	4310	Josh Berman

```
DROP TABLE Sales_and_Trading CASCADE;

CREATE TABLE IF NOT EXISTS Sales_and_Trading (
        DID int NOT NULL UNIQUE,
        Divisionname varchar(255) NOT NULL,
        equityPositions int NOT NULL,
        fixedIncomePositions int NOT NULL,
        topTrader varchar(255) NOT NULL,
        PRIMARY KEY (DID)
);
```

Functional dependencies

DID → divisionName, equityPositions, fixedIncomePositions, topTrader

Private_Wealth_Management table

The final division of AlanCorp. is Private Wealth Management which is tasked with the job of managing high and ultra high net worth clients' monetary savings. The table shows how many assets under management (total client dollars managed) the division controls along with the top ranking team, how many vacation days they have taken and their highest net worth client.

Sample Data

did integer	divisionname character varying(255)	assetsundermanagement integer	topteam character varying(255)	vacationdaystaken integer	highestnetworthclient character varying(255)
1	Private Wealth Management	500000000	Labouseur Wealth Management	5	54260771

```
DROP TABLE Private_Wealth_Management CASCADE;

CREATE TABLE IF NOT EXISTS Private_Wealth_Management (
        DID int NOT NULL UNIQUE,
        Divisionname varchar(255) NOT NULL,
        assetsUnderManagement int NOT NULL,
        topTeam varchar(255) NOT NULL,
        vacationDaysTaken int NOT NULL,
        highestNetWorthClient varchar(255) NOT NULL,
        PRIMARY KEY (DID)
);
```

Functional dependencies

DID → divisionName, assetsUnderManagement, topTeam, vacationDaysTaken, highestNetWorthClient

Awards table

The awards table simply lays out the award ID (AID), the person who earned it (PID), their respective rank on the list of the certain award and the most recent year in which they won the award. This is a highly competitive business so winning any of these awards are crucial for advancement.

```
DROP TABLE Awards CASCADE;

CREATE TABLE IF NOT EXISTS Awards (
        AID int NOT NULL,
        PID int NOT NULL,
        rank int NOT NULL,
        yearWon int NOT NULL,
        PRIMARY KEY (AID, PID, yearWon),
        FOREIGN KEY (AID) REFERENCES Accolades (AID),
        FOREIGN KEY (PID) REFERENCES People (PID)
);
```

Functional dependencies

AID, PID, Rank → YearWon

aid integer	pid integer	rank integer	yearwon integer
1	17	32	2015
1	17	24	2017
2	7	25	2015
3	2	17	2002
4	7	36	2007
4	13	48	2007
5	19	9	2015
6	5	37	2009
7	9	3	2013
7	18	5	2013
8	14	13	2014
9	12	21	2017
10	11	2	2001
10	11	4	2008
11	1	22	1998
12	20	19	1997
12	20	13	2007

Sample Data

Accolades table

The accolades table further discusses and describes the awards won by employees of the firm. In this table, the award ID (AID) is linked to the awards table to describe the name of the respective accolade along with the publication that issued the award.

```
DROP TABLE Accolades CASCADE;

CREATE TABLE IF NOT EXISTS Accolades (
        AID int NOT NULL UNIQUE,
        name varchar(255) NOT NULL,
        publication varchar(255) NOT NULL,
        PRIMARY KEY (AID)
);
```

Sample Data

aid integer	name character varying(255)	publication character varying(255)
1	Top 50 U.S. Money Managers	Investors Business Daily
2	Top 25 Global Traders	Bloomberg
3	Top 20 Most Profitable Acquisitions Made	Wall Street Journal
4	Top 50 Global Traders	ZeroHedge
5	Top 10 Connecticut Traders	Time
6	Top 50 Most Profitable Mergers Made	Investopedia
7	Top 5 Global Money Managers	Money Magazine
8	Top 15 Largest Money Managers	The Economist
9	Top 25 Most Influential Money Managers	Kiplingers
10	Top 5 Most Capital Raised	Barrons
11	Top 25 Global Traders	Fortune
12	Top 20 Most Profitable Mergers Made	Worth

Functional dependencies

AID → name, publication

Locations table

The final table in this relational database is the locations table. This better groups the three divisions into geographical locations where the respective office can be found. Specific information highlighted here includes the offices location ID (LID), state, city, street, address number and the number of floors in the building.

Functional dependencies

LID → state, city, street, addressNumber, numberOfFloors

```
DROP TABLE Locations CASCADE;

CREATE TABLE IF NOT EXISTS Locations (
        LID int NOT NULL UNIQUE,
        state varchar(255) NOT NULL,
        city varchar(255) NOT NULL,
        street varchar(255) NOT NULL,
        addressNumber int NOT NULL,
        numberOfFloors int NOT NULL,
        PRIMARY KEY (LID)
);
```

Sample Data

lid integer	state character varying(255)	city character varying(255)	street character varying(255)	addressnumber integer	numberoffloors integer
1	NY	New York	5th Avenue	522	50
2	CT	Greenwich	St. Adams	537	25
3	MA	Boston	Commonwealth Avenue	129	35

Views

firstname character varying(255)	lastname character varying(255)	position character varying(255)	numberoflanguagesspoken integer	cellphonenumber integer	emailssentinaday integer
Greg	King	Client Service Associate	3	51631856	30
Jessica	Albert	Client Service Associate	2	64951324	50
Dietrich	Mosel	Client Service Associate	4	65385109	34
Sam	Chantly	Client Service Associate	1	54675638	45
Emily	Roberts	Client Service Associate	1	93756475	25
Michaela	Murray	Client Service Associate	2	66254091	66
Bianca	Luparello	Client Service Associate	3	29061534	89

CSA_Personnel View: lists the employees that work as Client Service Associates across the three divisions (PWM, S&T and IB)

```
DROP VIEW CSA_Personnel CASCADE;
CREATE VIEW CSA_Personnel AS
SELECT p.firstname, p.lastname, c.position, c.numberOfLanguagesSpoken, cellPhoneNumber, emailsSentInADay
FROM people p, Client_Service_Associates c
WHERE p.PID= c.CSAID;
SELECT *
FROM CSA_Personnel
```

Views Continued…

Honors view:

lists the employee by first name and last name that won an award with the name of the respective award stated along with their rank on that list and the year won

```
DROP VIEW Honors CASCADE;
CREATE VIEW Honors AS
SELECT p.firstname, p.lastname, l.name, a.rank, a.yearWon
FROM people p, Awards a, Accolades l
WHERE p.PID = a.PID AND l.AID = a.AID;
SELECT *
FROM Honors
```

firstname character varying(255)	lastname character varying(255)	name character varying(255)	rank integer	yearwon integer
John	Lee	Top 50 U.S. Money Managers	32	2015
John	Lee	Top 50 U.S. Money Managers	24	2017
Alan	Labouseur	Top 25 Global Traders	25	2015
Mitch	Krout	Top 20 Most Profitable Acquisitions Made	17	2002
Vinny	Donatacci	Top 50 Global Traders	48	2007
Alan	Labouseur	Top 50 Global Traders	36	2007
Olivia	Cray	Top 10 Connecticut Traders	9	2015
Angela	Paulson	Top 50 Most Profitable Mergers Made	37	2009
Chris	Badolato	Top 5 Global Money Managers	3	2013
Kyle	Hannafin	Top 5 Global Money Managers	5	2013
Kenny	Walshek	Top 15 Largest Money Managers	13	2014
Mark	Valentino	Top 25 Most Influential Money Managers	21	2017
Gianna	Louro	Top 5 Most Capital Raised	2	2001
Gianna	Louro	Top 5 Most Capital Raised	4	2008
Jesse	Korris	Top 25 Global Traders	22	1998
Tom	Slattery	Top 20 Most Profitable Mergers Made	19	1997
Tom	Slattery	Top 20 Most Profitable Mergers Made	13	2007

Reports

```
SELECT city
FROM Locations
WHERE state NOT IN (SELECT state
                    FROM Locations
                    WHERE state = 'NY'
                    )
```

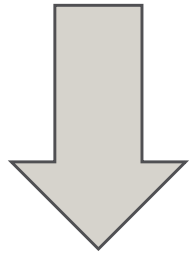
	city character varying(255)
1	Greenwich
2	Boston

This report shows the locations of offices, specifically cities, that are not in the state of New York

Reports Continued…

This specific query concentrates on the people table which is the most necessary to the proper functioning of the database. In this example, we can see which employees from the people table have more than 15 years of experience at the firm regardless of their division. Along with their first and last names, the result also shows the respective employees' hobby… of which Gianna has none because she is so consumed by her work!

```
SELECT firstname, lastname, hobby
FROM people
WHERE yearsOfExperience > 15;
```



firstname character varying(255)	lastname character varying(255)	hobby character varying(255)
Jesse	Korris	Skiing
Mitch	Krout	Hangliding
Chris	Badolato	Restoring Cars
Gianna	Louro	
John	Lee	Video Games
Kyle	Hannafin	Basketball
Olivia	Cray	Coding

Stored Procedures

```
CREATE OR REPLACE FUNCTION Division_Location() RETURNS trigger AS
$$
BEGIN
        IF NEW.PID = (SELECT MAX(PID) FROM People) THEN
                IF NEW.DID = 1 THEN
                        UPDATE People
                        SET LID = 3
                        WHERE PID = NEW.PID;
                END IF;

                IF NEW.DID = 2 THEN
                        UPDATE People
                        SET LID = 2
                        WHERE PID = NEW.PID;
                END IF;

                IF NEW.DID = 3 THEN
                        UPDATE People
                        SET LID = 1
                        WHERE PID = NEW.PID;
                END IF;

        END IF;
        RETURN NEW;

END;
$$
LANGUAGE PLPGSQL;

DROP TRIGGER IF EXISTS Divison_Location ON People;
CREATE TRIGGER Division_Location
AFTER INSERT ON People
FOR EACH ROW
EXECUTE PROCEDURE Division_Location();
```

Division_Location:

This stored procedure, relative to the trigger that creates it, is based on the people, divisions and locations tables. All three tables are utilized to get a better grasp on what person works in what division and where. Values are assigned based on the divisions specific geographic location and the stored procedure checks through the trigger to make sure each employee is assigned to the proper division and its relative location.

Stored Procedures

```
CREATE OR REPLACE FUNCTION Pay_Raise() RETURNS trigger AS
$$
BEGIN
        IF OLD.numberOfClients < NEW.numberOfClients THEN
                IF (NEW.numberOfClients - OLD.numberOfClients / OLD.numberOfClients) >= .20 THEN
                        UPDATE Managing_Directors
                        SET salary = salary + 50000
                        WHERE MID = NEW.MID;
                END IF;

        END IF;
        RETURN NEW;

END;
$$
LANGUAGE PLPGSQL;

CREATE TRIGGER Pay_Raise
AFTER UPDATE ON Managing_Directors
FOR EACH ROW
EXECUTE PROCEDURE Pay_Raise();
```

Pay_Raise:

This stored procedure concentrates on the Managing_Directors table as does the trigger it is associated with (pictured here). Basically, it automates a growth in client base of 20% with a $50,000 increase of that Managing Directors salary. The stored procedure checks every row and executes at the end.

Stored Procedures

```
CREATE OR REPLACE FUNCTION Average_Rank (int) RETURNS int AS
$$
DECLARE

        AID int = $1;

BEGIN
        RETURN AVG(Rank)
                FROM Awards
                WHERE Awards.AID = $1;

END;
$$
LANGUAGE PLPGSQL;
```

average_rank integer	average_rank integer
42	28

```
SELECT Average_Rank(4), Average_Rank(1)
```

Average_Rank:

This stored procedure focuses on the Awards table specifically. It returns the average rank for a certain employee if they were to win multiple awards. For example, the average rank of employees who have earned awards 1 and 4 are 28/50 and 42/50 respectively.

Triggers

```
CREATE OR REPLACE FUNCTION Division_Location() RETURNS trigger AS
$$
BEGIN
        IF NEW.PID = (SELECT MAX(PID) FROM People) THEN
                IF NEW.DID = 1 THEN
                        UPDATE People
                        SET LID = 3
                        WHERE PID = NEW.PID;
                END IF;

                IF NEW.DID = 2 THEN
                        UPDATE People
                        SET LID = 2
                        WHERE PID = NEW.PID;
                END IF;

                IF NEW.DID = 3 THEN
                        UPDATE People
                        SET LID = 1
                        WHERE PID = NEW.PID;
                END IF;

        END IF;
        RETURN NEW;

END;
$$
LANGUAGE PLPGSQL;

CREATE TRIGGER Division_Location
AFTER INSERT ON People
FOR EACH ROW
EXECUTE PROCEDURE Division_Location();
```

Division_Location:

If you move locations, this change will be reflected in the division in which employees are stationed. As of now, The Private Wealth Management division is classified as division 1 with a location ID of 3. Sales and Trading has a division ID of 2 and a location ID of 2. Investment Banking has a division ID of 3 and a location ID of 1. If a new employee were to be hired, it would assign them a relative DID according to the LID and vice versa as they are dependent on each other.

Triggers

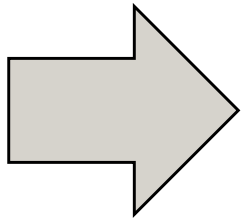
```
CREATE OR REPLACE FUNCTION Pay_Raise() RETURNS trigger AS
$$
BEGIN
        IF OLD.numberOfClients < NEW.numberOfClients THEN
                IF (NEW.numberOfClients - OLD.numberOfClients / OLD.numberOfClients) >= .20 THEN
                        UPDATE Managing_Directors
                        SET salary = salary + 50000
                        WHERE MID = NEW.MID;
                END IF;

        END IF;
        RETURN NEW;

END;
$$
LANGUAGE PLPGSQL;

CREATE TRIGGER Pay_Raise
AFTER UPDATE ON Managing_Directors
FOR EACH ROW
EXECUTE PROCEDURE Pay_Raise();
```

Pay_Raise:

This specific trigger focuses on the Managing Directors table specifically regarding their salaries. If a managing director were to obtain 20% more clients than their current base, a $50,000 boost to their salary would be added. This trigger will update the respective managing directors salary to reflect this increase in clients.

Pay raise before and after…

mid integer	position character varying(255)	salary integer	numberofclients integer	meetingsperyear integer
1	Managing Director	250778	275	223
9	Managing Director	176348	236	187
11	Managing Director	326331	401	356
17	Managing Director	108450	210	142
19	Managing Director	223791	260	218
20	Managing Director	223791	260	218



mid integer	position character varying(255)	salary integer	numberofclients integer	meetingsperyear integer
1	Managing Director	250778	275	223
9	Managing Director	176348	236	187
11	Managing Director	326331	401	356
17	Managing Director	108450	210	142
20	Managing Director	223791	260	218
19	Managing Director	273791	315	218

Security and Privileges

System Administrator

```
CREATE ROLE system administrator;
GRANT ALL ON ALL TABLES
SCHEMA PUBLIC
To system administrator;
```

Board of Directors

```
CREATE ROLE board of directors
GRANT SELECT ON Divisions, Private_Wealth_Management, Sales_and_Trading,
                                    Investment_Banking
TO board of directors;
GRANT UPDATE ON Divisions, Private_Wealth_Management, Sales_and_Trading,
                                    Investment_Banking
```

Security and Privileges

Compensation Committee
CREATE ROLE audit committee
GRANT SELECT ON People, Client_Service_Associates, Managing_Directors, Analysts

TO compensation committee;
GRANT UPDATE ON People, Client_Service_Associates, Managing_Directors, Analysts

Employees
CREATE ROLE employees
GRANT SELECT ON People, Client_Service_Associates, Managing_Directors, Analysts, Awards

TO Employees;
GRANT UPDATE ON People

Implementation Notes

There were definitely some issues I came across when trying to execute some triggers associated with the awards table specifically, which will be discussed in the known problems section of this report. However, this database does a good job of displaying the overall resources of AlanCorp. Broken down by people, a.k.a. employees, that work in three different capacities, we find that there are also three divisions in which they can work. These are Private Wealth Management, Investment Banking and Sales and Trading. Going off of that, we can find these offices according to the locations table and awards are assigned based upon the person working in a certain office. An award is given by a certain publication of which there is a list where the certain employee is ranked according to their performance in that business. The relationships developed stem from the people table which is the centralized location for this database. Branching from people are many sub-entities of which there can be more information. It would be nice to display employees who work in certain roles and the progression at which they moved up the corporate ladder. In addition, finding a way to add more than one row to each division table would be nice for the future as the business will only expand from here.

Known Problems

```
NOTICE:  trigger "divison_location" for relation "people" does not exist, skipping
ERROR:  stack depth limit exceeded
HINT:  Increase the configuration parameter "max_stack_depth" (currently 2048kB), after ensuring the platform's stack depth limit is adequate.
CONTEXT:  SQL statement "SELECT 1 FROM ONLY "public"."divisions" x WHERE "did" OPERATOR(pg_catalog.=) $1 FOR KEY SHARE OF x"
SQL statement "UPDATE People
            SET LID = 2
            WHERE PID = NEW.PID"
PL/pgSQL function division_location() line 11 at SQL statement
SQL statement "UPDATE People
            SET LID = 2
            WHERE PID = NEW.PID"
PL/pgSQL function division_location() line 11 at SQL statement
SQL statement "UPDATE People
```

One of the problems I encountered in trying to execute a trigger for updating a change in the location of an employee. When attempting to insert and properly implement the trigger, I received an error message that read “max_stack_depth.” I tried to solve this by creating a second trigger so that there would only be one trigger active at a time in an effort to reduce the amount of space it was utilizing. In the future, it would be a good idea to implement a trigger so that an employees change in location can be automatically updated instead of manually which will take more time.

Future Enhancements

There is always more that can be done and improved, especially related to databases and ones associated with constantly changing entities like financial services corporations. In reference to AlanCorp's database, the business will be looking to expand into new market segments in the coming years and hire more employees as the economy starts to ramp up. New employee roles will have to be created along with divisions as the company seeks to diversify their respective offerings to clients. Going forward, the locations table will need to be updated since the firm will also expand in a geographic sense. However, as the company looks to make major changes, it would be smart to first identify and fix problems with the current database as there are a few. Being able to automate more tasks instead of manually updating tables is crucial for the success of the firm as things will be constantly changing and it would be a waste of resources and time to update tables on a day-to-day basis.