Gladstone Stoneworks

Joseph Archer

December 2, 2013

Table of Contents

Table Of Contents---------- Page 2

Executive Summary---------- Page 3

Entity Relationship Diagram---------- Page 4

Table: People ---------- Page 5

Table: Clients ---------- Page 6

Table: Employees ---------- Page 7

Table: Classifications ---------- Page 8

Table: Projects ---------- Page 9

Table: Materials ---------- Page 10

Table: MaterialTypes ---------- Page 11

Table: Suppliers ---------- Page 12

Table: Employees_Classifications ---------- Page 13

Table: Clients_Projects---------- Page 14

Table: Employees_Projects ---------- Page 15

Table: Materials_Type ---------- Page 16

Table: Projects_Materails ---------- Page 17

Table: Suppliers_Materials ---------- Page 18

View: CURRENT_EMPLOYEES ---------- Page 19

View: CURRENT_INVENTORY ---------- Page 20

Report: Order History ---------- Page 21

Report: Project History---------- Page 22

Stored Procedure: CalculateAge()---------- Page 23

Trigger: Check_Project ---------- Page 24

Trigger:Check_Inventory---------- Page 25

Security: Administrator:---------- Page 26

Security: Assistant---------- Page 27

Security:Project Manager---------- Page 28

Security:Basic_Employee---------- Page 29

Implementation Notes---------- Page 30

Known Problems---------- Page 31

Future Enhancements---------- Page 32

Executive Summary

The goal of this database is to help with the day to day business needs of Gladstone Stoneworks, a stone masonry company located in Gladstone, New Jersey. The company is in need of a database system that will help to manage their Projects , Employees , and Materials efficiently. The database will also needs to store information about clients. The database has three main security roles helping to ensure that private records are not able to be seen by nosey employees and uses triggers to prevent the Inventory System from falling below zero.

Entity Relationship Diagram

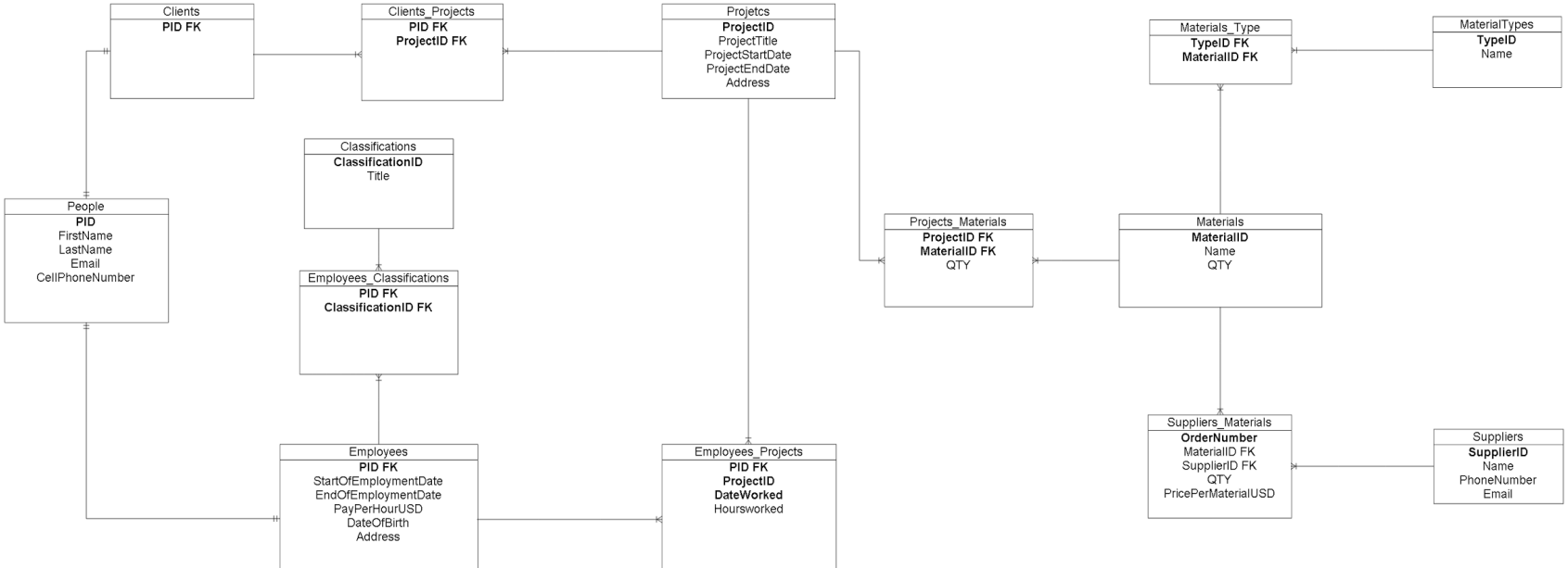

Table: People

```
CREATE TABLE IF NOT EXISTS     People
(
PID                    SERIAL           NOT NULL ,
FirstName              VARCHAR(255) NOT NULL ,
LastName               VARCHAR(255) NOT NULL ,
Email                  VARCHAR(255) NOT NULL ,
CellPhoneNumber        CHAR(12)         NOT NULL ,
Primary Key(PID)
) ;
```

Functional Dependencies

PID → (FirstName , LastName , Email , CellPhoneNumber)

The purpose of the People table is to hold Information that both Clients and Employees can have.

	pid [PK] serial	firstname character varying(255)	lastname character varying(255)	email character varying(255)	cellphonenumber character(12)
1	1	Joe	Archer	Joseph.Archer1@Marist.edu	908-500-2877
2	2	Donald	Lynn	donallynn@webmine.com	908-546-7878
3	3	Bobby	Blanco	boblanco@freespace.com	908-876-4545
4	4	Arnon	Teegarden	arnon_teeg@yahoo.com	908-678-4567
5	5	Gregg	Holguin	gr_holgui@infoseller.com	908-451-7890
6	6	Neal	Wilmot	nea-wilmot@hotmail.com	909-678-2312
7	7	Lewis	Chou	lewi-cho@webmine.com	908-453-9087
8	8	Eduardo	Blanks	eduar_blan@google.com	908-720-1212
9	9	Charles	Flock	chafl@freespace.com	908-567-7878
10	10	Janice	Darlington	jani_da@yoohoo.com	908-657-8664
11	11	Denise	Grenier	denise_gre@google.com	908-445-7890
12	12	Karina	Brigman	karina_br@google.com	908-977-5656
13	13	Aretina	Stiles	areti.st@linux.com	908-636-2847
14	14	Shay	Harrell	shay-harrell@mail.com	908-345-9090
15	15	Earl	Hubbard	earlhubbard@mail.com	908-678-2340
*					

Table: Clients

```
CREATE TABLE IF NOT EXISTS     Clients
(
PID         INT  NOT NULL references People(PID) ,
Primary Key(PID)
) ;
```

The purpose of the Clients table is to store a PID (Person ID).

Functional Dependencies

PID →

	pid [PK] integer
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
*	

Table: Employees

```
CREATE TABLE IF NOT EXISTS     Employees
(
PID                            INT              NOT NULL references People(PID)     ,
StartOfEmploymentDate          DATE             NOT NULL                            ,
EndOfEmploymentDate            DATE                                                 ,
PayPerHourUSD                  MONEY            NOT NULL                            ,
DateOfBirth                    DATE             NOT NULL                            ,
Address                        VARCHAR(255) NOT NULL                                ,
Primary Key(PID)
) ;
```

The purpose of the Employees table is to store information about employees both past and current.

Functional Dependencies

PID → (StartOfEmploymentDate , EndOfEmploymentDate , PayPerHourUSD , DateOfBirth , Address)

	pid [PK] ir	startofemploy date	endofemployi date	payperhouru money	dateofbirth date	address character varying(255)
1	9	1999-12-01	2003-04-19	$25.00	1960-05-13	2251 Tawny Lane, Mad Creek, New Jersey, 07189-8749
2	10	1995-09-12		$40.00	1968-07-08	9681 Cotton Grove Route, Tater Peeler, New Jersey, 08474-0577
3	11	1991-10-11		$50.00	1970-09-08	4276 Lost Butterfly Autoroute, Eau Claire, New Jersey, 07087-3039
4	12	1991-10-11		$50.00	1970-09-08	8151 Sunny Sky Cove, Bee Tree, New Jersey, 07066-6198
5	13	2002-02-04		$25.00	1981-03-08	4687 Pleasant Common, Oatmeal, New Jersey, 07658-2524
6	14	2005-10-15		$25.00	1984-10-12	8906 Merry Manor, Truth Home, New Jersey, 08825-5260
7	15	1998-07-11		$35.00	1974-04-20	4209 Rustic Forest Field, Fivemile, New Jersey, 07837-7712
*						

Table: Classifications

```
CREATE TABLE IF NOT EXISTS     Classifications
(
ClassificationID     SERIAL           NOT NULL    ,
Title                VARCHAR(20)  NOT NULL    ,
Primary Key (ClassificationID)
) ;
```

The purpose of the Classifications table is store titles for the types of jobs that employees can have.

Functional Dependencies

ClassificationID → (Title)

	classificationi [PK] serial	title character varying(2
1	1	Laborer
2	2	Office Worker
3	3	Iron Worker
4	4	Estimator
5	5	Carpenter
6	6	Painter
7	7	Project Manager
*		

Table: Projects

```
CREATE TABLE IF NOT EXISTS     Projects
(
ProjectID            SERIAL             NOT NULL     ,
ProjectTitle         VARCHAR(255)  NOT NULL     ,
ProjectStartDate  Date                  NOT NULL     ,
ProjectEndDate    Date                        NULL     ,
Address              VARCHAR(255)  NOT NULL     ,
Primary Key(ProjectID)
) ;
```

The purpose of the Projects table is store information about projects both past and current.

Functional Dependencies

ProjectID → (ProjectTitle , ProjectStartDate , ProjectEndDate , Address)

	project [PK] se	projecttitle character varying(:	projectstartd date	projectendda date	address character varying(255)
1	1	Gladstone Bank	2000-09-12	2002-09-29	3891 Honey Willow Estates, Buttzville, New Jersey, 08234-6231
2	2	The Barns	2002-10-03	2007-12-12	3773 Dewy Chase, Hoodoo, New Jersey, 08381-1634
3	3	Pizza Como	2009-12-01		6072 Silver Carrefour, Nutt, New Jersey, 07786-2210
4	4	Zubrow	2010-06-14		8306 Cozy Pines, Summertime, New Jersey, 07968-1927, US
5	5	Gladestone Park	2013-12-01		401 Crystal Lake Island, Tango, New Jersey, 08955-5876, US
*					

Table: Materials

```
CREATE TABLE IF NOT EXISTS    Materials
(
MaterialID  SERIAL              NOT NULL    ,
Name        VARCHAR(255)   NOT NULL    ,
QTY         INT                 NOT NULL    ,
Primary Key(MaterialID)
) ;
```

The Purpose of the Materials table is store information about the names of all the different kinds of materials and the Materials table is also used to calculate the CURRENT_INVENTORY View.

Functional Dependencies

MaterialID → (Name , QTY)

	materialid [PK] serial	name character vary	qty integer
1	1	Solid Brick	500
2	2	Cored Brick	500
3	3	Granite	500
4	4	Marble	500
5	5	Blue Stone	500
6	6	Red Stone	500
7	7	Sandstone	500
8	8	Lime	500
9	9	Sand	500
10	10	Cement	500
*			

Table: MaterialTypes

```
CREATE TABLE IF NOT EXISTS  MaterialTypes
(
TypeID      SERIAL              NOT NULL ,
Name        VARCHAR(255) NOT NULL ,
Primary Key (TypeID)
) ;
```

The purpose of the MaterialTypes table is to store the names of the different types of materials that Gladstone Stoneworks uses.

Functional Dependencies

TypeID → (Name)

	typeid [PK] serial	name character var
1	1	Stone
2	2	Rock
3	3	Brick
4	4	Ingredient
*		

Table: Suppliers

```
CREATE TABLE IF NOT EXISTS Suppliers
(
SupplierID      SERIAL          NOT NULL ,
Name            VARCHAR(255)    NOT NULL ,
PhoneNumber     VARCHAR(255)    NOT NULL ,
Email           VARCHAR(255)    NOT NULL ,
Primary Key (SupplierID)
) ;
```

The purpose of the Suppliers table is to store information about the Suppliers and provide contact information.

Functional Dependencies

SupplierID → (Name , PhoneNumber , Email)

	supplie [PK] se	name character varying(255	phonenumber character varying	email character varying(255)
1	1	The Rock Company	609-585-5000	RockCompany@gmail.com
2	2	Athenia Mason Supp.	973-253-0570	masonsupply@gmail.com
3	3	The Stone People	908-456-1231	StonePeople@gmail.com
*				

Table: Employees_Classifications

```
CREATE TABLE IF NOT EXISTS Employees_Classifications
(
PID                 INT NOT NULL references Employees(PID)                  ,
ClassificationID    INT NOT NULL references Classifications(ClassificationID) ,
Primary Key (PID , ClassificationID)
) ;
```

The Purpose of The Employees_Classifications table is to connect the Employees and Classifications table. This table allows employees to be given titles.

Functional Dependencies

PID → (ClassificationID)

	pid [PK] integer	classificationi [PK] integer
1	9	1
2	10	2
3	11	3
4	12	4
5	13	5
6	14	1
7	15	7
*		

Table: Clients_Projects

```
CREATE TABLE IF NOT EXISTS Clients_Projects
(
PID             INT NOT NULL references Clients(PID)          ,
ProjectID       INT NOT NULL references Projects(ProjectID)   ,
Primary Key (PID , ProjectID )
);
```

The purpose of the Clients_Projects table is to connect the Clients and the Projects table. This table allows clients to be connected with a Project.

Functional Dependencies

(PID , ProjectID) →

	pid [PK] integer	projectid [PK] integer
1	1	1
2	2	2
3	3	3
4	4	4
5	5	5
*		

Table: Employees_Projects

```
CREATE TABLE IF NOT EXISTS Employees_Projects
(
PID             INT     NOT NULL references Employees(PID)      ,
ProjectID       INT     NOT NULL references Projects(ProjectID) ,
DateWorked    DATE NOT NULL                                     ,
HoursWorked  INT     NOT NULL                                   ,
Primary Key ( PID , ProjectID , DateWorked)
) ;
```

The purpose of the Employees_Projects table is to connect the Employees and Projects table. This table is used to track hours worked and days worked for workers.

Funcitonal Dependencies

(PID , ProjectID , DateWorked) → HoursWorked

	pid [PK] integer	projectid [PK] integer	dateworked [PK] date	hoursworked integer
1	10	1	2000-10-12	8
2	10	1	2000-10-13	8
3	10	1	2000-10-14	8
4	10	3	2007-10-12	8
5	11	2	2003-04-17	8
6	11	2	2003-04-18	8
7	12	3	2000-10-12	8
*				

Table: Materials_Type

```
CREATE TABLE IF NOT EXISTS Materials_Type
(
MaterialID     INT NOT NULL references Materials(MaterialID)     ,
TypeID         INT NOT NULL references MaterialTypes(TypeID) ,
Primary Key ( MaterialID , TypeID )
) ;
```

The purpose of the Materials_Type table is to connect the Materials and MaterialTypes Table. This table allows Materials to be given a type such as stone or ingredient.

Functional Dependencies

(MaterialID , TypeID) →

	materialid [PK] integer	typeid [PK] integer
1	1	3
2	2	3
3	3	2
4	4	2
5	5	1
6	6	1
7	7	1
8	8	4
9	9	4
10	10	4
*		

Table: Projects_Materials

```
CREATE TABLE IF NOT EXISTS Projects_Materials
(
ProjectID     INT NOT NULL references Projects(ProjectID)     ,
MaterialID  INT NOT NULL references Materials(MaterialID) ,
QTY             INT NOT NULL                                          ,
Primary Key (ProjectID , MaterialID )
) ;
```

The purpose of the Projects_Materials table is to connect the Projects and Materials tables. This table is used to calculate Current_Inventory as well as to show what materials have been used on what projects.

Functional Dependencies

(ProjectID , MaterialID) → QTY

	projectid [PK] integer	materialid [PK] integer	qty integer
1	1	8	100
2	1	9	100
3	2	5	100
4	2	6	100
5	2	7	100
6	3	1	100
7	3	2	100
8	3	3	100
9	3	4	100
10	4	1	1
*			

Table: Suppliers_Materials

```
CREATE TABLE IF NOT EXISTS  Suppliers_Materials
(
OrderNumber            SERIAL   NOT NULL                                    ,
MaterialID             INT      NOT NULL references Materials(MaterialID)  ,
SupplierID             INT      NOT NULL references Suppliers(SupplierID)  ,
QTY                    INT      NOT NULL                                    ,
PricePerMaterialUSD    MONEY    NOT NULL                                    ,
Primary Key (OrderNumber)
) ;
```

The purpose of Suppliers_Materials is to connect the Suppliers and Materials tables. This table is used to track Orders from suppliers.

Functional Dependencies

OrderNumber → (MaterialID , SupplierID , QTY , PricePerMaterialUSD)

	ordernumber [PK] serial	materialid integer	supplierid integer	qty integer	pricepermate money
1	1	1	1	50	$2.00
2	2	2	2	50	$5.00
3	3	3	3	50	$2.50
4	4	4	3	50	$2.50
*					

View: CURRENT_EMPLOYEES

```
CREATE VIEW CURRENT_EMPLOYEES AS
SELECT FirstName , LastName , Title , PayPerHourUSD
FROM Employees_Classifications , Employees , People , Classifications
WHERE People.PID = Employees.PID
AND Employees.PID = Employees_Classifications.PID
AND Classifications.ClassificationID = Employees_Classifications.ClassificationID
AND Employees.EndofEmploymentDate IS NULL;
```

The Purpose of the Current_Employees view is to show every current employee at Gladstone Stoneworks as well as some information about them such as their title and age.

	firstname character var	lastname character var	title character varying(20)	age integer	payperhourus money
1	Janice	Darlington	Office Worker	45	$40.00
2	Denise	Grenier	Iron Worker	43	$50.00
3	Karina	Brigman	Estimator	43	$50.00
4	Aretina	Stiles	Carpenter	32	$25.00
5	Shay	Harrell	Laborer	29	$25.00
6	Earl	Hubbard	Project Manager	39	$35.00

View: CURRENT_INVENTORY

```
CREATE VIEW CURRENT_INVENTORY AS

SELECT Items.ItemID as TEST , Items.Name , Items.QTY - a.ProjectQTY + b.SupplierQTY as QTY

FROM Items , (SELECT Projects_Items.ItemID , SUM (Projects_Items.QTY ) as ProjectQTY FROM Projects_Items GROUP BY Projects_Items.ItemID)  as a , (SELECT Suppliers_Items.ItemID , SUM (Suppliers_Items.QTY ) as SupplierQTY FROM Suppliers_Items GROUP BY Suppliers_Items.ItemID) as b

WHERE a.ItemID = Items.ItemID

AND b.ItemID = Items.ItemID

GROUP BY TEST, a.ProjectQTY , b.SupplierQTY ;
```

	id integer	name character varyi	qty bigint
1	1	Solid Brick	449
2	2	Cored Brick	450
3	3	Granite	450
4	4	Marble	450

The purpose of the Current_Inventory view is to show what kinds of items the company has available as well as how many of each one.

Report: Order History

```
SELECT Suppliers_Materials.OrderNumber , Materials.Name , Suppliers.Name , Suppliers_Materials.QTY , PricePerMaterialUSD , Suppliers_Materials.QTY *
PricePerMaterialUSD AS  OrderTotal

From Materials , Suppliers , Suppliers_Materials

Where Materials.MaterialID = Suppliers_Materials.MaterialID

AND Suppliers.SupplierID = Suppliers_Materials.SupplierID

ORDER BY OrderNumber ASC
```

The purpose of the Order History Report is to show ever order from the company , the cost of the order , and other important information about the transaction.

	ordernumber integer	name character varying(255)	name character varying(255)	qty integer	pricepermaterialusd money	ordertotal money
1	1	Solid Brick	The Rock Company	50	$2.00	$100.00
2	2	Cored Brick	Athenia Mason Suppl	50	$5.00	$250.00
3	3	Granite	The Stone People	50	$2.50	$125.00
4	4	Marble	The Stone People	50	$2.50	$125.00

Report: Project History

```
Select ProjectTitle , Materials.Name , Projects_Materials.QTY
from Projects_Materials , Materials , Projects
WHERE Materials.MaterialID = Projects_Materials.MaterialID
and Projects.ProjectID = Projects_Materials.ProjectID
and Projects.ProjectID = '--';
```

The purpose of the Project History Report is to show what kinds of materials a certain project has used.

NOTE: See Implementation Notes for more Information about the Project History Report

	projecttitle character varying(255)	name character varying(255)	qty integer
1	Pizza Como	Solid Brick	100
2	Pizza Como	Cored Brick	100
3	Pizza Como	Granite	100
4	Pizza Como	Marble	100

Stored Procedure: CalculateAge()

```
CREATE OR REPLACE FUNCTION CALCULATE_AGE(EmployeeID INTEGER) RETURNS Integer AS $$
DECLARE
        Age Integer := (SELECT EXTRACT (YEAR FROM AGE(CURRENT_DATE , (SELECT DateofBirth From Employees where Employees.pid = employeeID) ) ) );
BEGIN
        RETURN age ;
END ;
$$ LANGUAGE plpgsql;
```

The purpose of CalculateAge() procedure is to calculate a persons age. This procedure is used for the Current_Employees view to show each employees age.

Trigger: Check_Project

```
CREATE OR REPLACE FUNCTION Check_Project() RETURNS TRIGGER AS $Check_Project$
    BEGIN
        IF
          (SELECT ProjectEndDate
          FROM Projects
          WHERE ProjectID = NEW.ProjectID)
           IS  NOT NULL THEN
              RAISE EXCEPTION 'Cannot insert an employee into a project that is already finished' ;
          END IF;
      RETURN NEW;
    END;
$Check_Project$ LANGUAGE plpgsql ;


CREATE TRIGGER Check_Project
   BEFORE INSERT ON Employees_Projects
    FOR EACH ROW
    EXECUTE PROCEDURE Check_Project() ;
```

The Purpose of the Check_Project trigger is to prevent someone from entering a record into the Employees_Projects table for a project that has already be completed. This trigger throws an exception if the Project has already been completed.

Trigger: Check_Inventory

```
CREATE OR REPLACE FUNCTION Check_Inventory() RETURNS TRIGGER AS $Check_Inventory$
    DECLARE
        A INTEGER := (SELECT QTY FROM CURRENT_INVENTORY WHERE ID IN ( SELECT Materials.MaterialID From Materials WHERE Materials.MaterialID =
New.MaterialId) ) ;
        B INTEGER := New.QTY ;
        C  INTEGER :=  A - B;
BEGIN
            IF (C  < 0) THEN
                                RAISE EXCEPTION 'Not Enough In Inventory';
                        END IF;
              RETURN NEW;
            END;
$Check_Inventory$ LANGUAGE plpgsql;


CREATE TRIGGER Check_INVENTORY
    BEFORE INSERT ON Projects_Materials
    FOR EACH ROW EXECUTE PROCEDURE Check_Inventory() ;
```

The Purpose of the Check_Inventory Trigger is to prevent the Inventory from dropping below zero. This trigger is called before every insert of the Projects_Materials table.

Security: Administrator

The Administrator role has the highest level of access to the database. The Administrator is not limited by any restrictions and has free reign over the database.

Security: Assistant

The Assistant role has the second highest access to the database. The Assistant role is used to do all inserts and updates in the database and is only restricted from accessing deleting records from certain tables.

```
REVOKE  DELETE ON Suppliers_Materials    FROM Assistant;
REVOKE  DELETE ON Projects               FROM Assistant;
REVOKE DELETE ON Employees_Projects      FROM Assistant;
```

Security: Project Manager

The Project Manager role has the second lowest access to the database. This role has access to select from more tables than the Basic_Employee.

```
REVOKE ALL PRIVILEGES ON People                    FROMProject_Manager;
REVOKE ALL PRIVILEGES ON Clients                   FROMProject_Manager;
REVOKE ALL PRIVILEGES ON Employees                 FROMProject_Manager;
REVOKE ALL PRIVILEGES ON Classifications           FROMProject_Manager;
REVOKE ALL PRIVILEGES ON Projects                  FROMProject_Manager;
REVOKE ALL PRIVILEGES ON Materials                 FROMProject_Manager;
REVOKE ALL PRIVILEGES ON MaterialTypes             FROMProject_Manager;
REVOKE ALL PRIVILEGES ON Suppliers                 FROMProject_Manager;
REVOKE ALL PRIVILEGES ON Employees_Classifications FROMProject_Manager;
REVOKE ALL PRIVILEGES ON Clients_Projects          FROMProject_Manager;
REVOKE ALL PRIVILEGES ON Employees_Projects        FROMProject_Manager;
REVOKE ALL PRIVILEGES ON Materials_Type            FROMProject_Manager;
REVOKE ALL PRIVILEGES ON Projects_Materials        FROMProject_Manager;
REVOKE ALL PRIVILEGES ON Suppliers_Materials       FROMProject_Manager;
```

```
GRANT  SELECT ON * FROM Projects;
GRANT  SELECT ON * FROM Projects_Materials;
GRANT  SELECT ON * FROM Employees;
GRANT SELECT ON * FROM Suppliers_Materials;
GRANT SELECT ON * FROM Employees_Classifications;
GRANT SELECT ON * FROM Classifications;
GRANT SELECT ON * FROM Materials_Type;
GRANT SELECT ON * FROM MaterialTypes;
GRANT SELECT ON * FROM Suppliers;
```

Security: Basic_Employee

The Basic_Employee role has the lowest level of access to the database. The Basic_Employee role is limited by many restrictions and is only allowed to select from certain tables.

```
REVOKE ALL PRIVILEGES ON People                      FROMBasic_Employee;
REVOKE ALL PRIVILEGES ON Clients                     FROMBasic_Employee;
REVOKE ALL PRIVILEGES ON Employees                   FROMBasic_Employee;
REVOKE ALL PRIVILEGES ON Classifications             FROMBasic_Employee;
REVOKE ALL PRIVILEGES ON Projects                    FROMBasic_Employee;
REVOKE ALL PRIVILEGES ON Materials                   FROMBasic_Employee;
REVOKE ALL PRIVILEGES ON MaterialTypes               FROMBasic_Employee;
REVOKE ALL PRIVILEGES ON Suppliers                   FROMBasic_Employee;
REVOKE ALL PRIVILEGES ON Employees_Classifications   FROMBasic_Employee;
REVOKE ALL PRIVILEGES ON Clients_Projects            FROMBasic_Employee;
REVOKE ALL PRIVILEGES ON Employees_Projects          FROMBasic_Employee;
REVOKE ALL PRIVILEGES ON Materials_Type              FROMBasic_Employee;
REVOKE ALL PRIVILEGES ON Projects_Materials          FROMBasic_Employee;
REVOKE ALL PRIVILEGES ON Suppliers_Materials         FROMBasic_Employee;
```

```
GRANT , SELECT ON * FROM Projects;
GRANT , SELECT ON * FROm Projects_Materials;
```

Implementation Notes

- When Entering a PhoneNumber use the format xxx-xxx-xxxx.

- When using the Project History report , the user must manually enter the projectID in the query for the project that they interested in.

Known Problems

- In the Employees_Projects table an employee can enter a date that is not between the ProjectStartDate and the ProjectEndDate

- The Projects_Materials Table does not keep track of a PricePerMaterialUsed. This means that the cost for each Project must be calculated outside of the database.

Future Enhancements

- Gladstone Stoneworks also rents machines and equipment from other companies. In the future a table for rentals should be included. This table will allow the company to track information about when rentals are due back and what rental company has the lowest price.

- Gladstone Stoneworks has three different yards available to them for storage of materials. In the future a few more tables will be added to show the current quantity of each material for each yard.

- Gladstone Stoneworks occasionally hires union workers , these workers are paid differently and are only allowed to work a certain amount of hours per week before they receive overtime. In the future, a feature will be added to sum the hours of any union worker for the current week.