PHASE 3: SUBMISSION CHECKLIST/SIGNOFF SHEET

GROUP#: 8 **GROUP NAME:**

Deliverables:

- Requirements Description
- ER Diagram with Min/Max Specifications
- ER Diagram Uncaptured Constraints
- Relational Schema with Referential Integrity
- Queries with brief description
- DDL statements
- Completed Oracle Implementation (Attached Team SQL script)

Assessment:

- Group Status Report

We have each reviewed the contents of this deliverable.

Phase Leader	Corey Gregory	_____
Phase Recorder	Scott Rowe	_____
Phase Checker	Adam Keech	_____
Technical Advisor	Steven Anderson	_____

Introduction:

This database is for a storage facility with multiple rental units, it is used by the managers and upper management to track and view pertinent data, including renting, taking payments from existing tenants, and managing status of storage units.

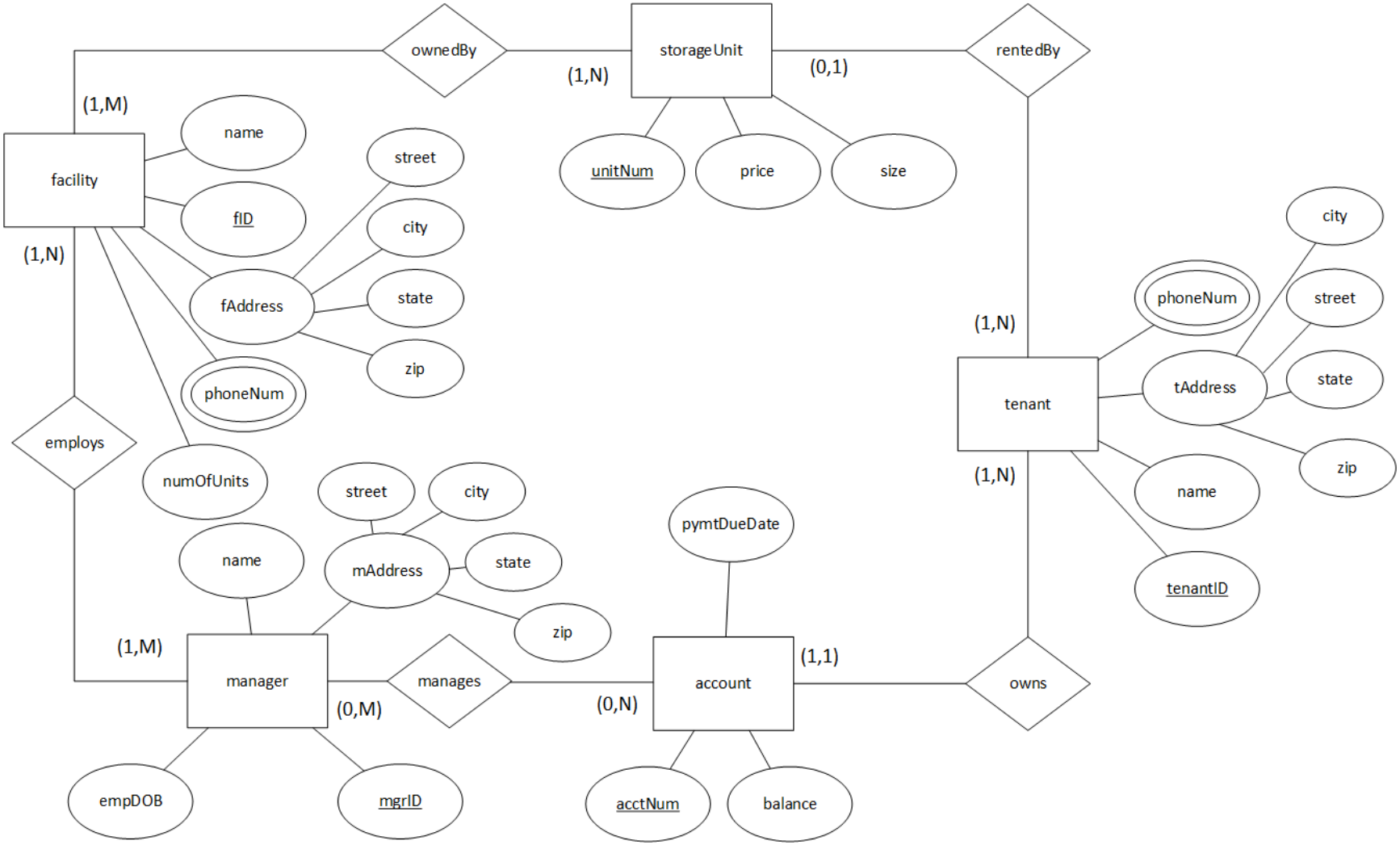
Requirements Description

The enterprise is a storage facility, the database contains information about storage units, their size, price per month, the tenants who rent them, and the managers that run the facility and handle tenant accounts.

- Each storage unit is given a letter and a number, for example, A1, A2, …, etc., progressing alphabetically the larger the unit size is and numerically for how many there are. Storage unit sizes typically range from (A) 5’x5’, (B) 5x10’, (C) 10’x10’, (D) 10’x15’, (E) 10’x20’. So, the fifth 5x10 would be B5. A unit can be vacant, or it can be rented. It can only be rented by one person.
- Each tenant has a phone number, an address, a name, and a tenant ID. Each tenant has an account which includes their account number, payment due date, and the balance that they pay each month. Tenants can rent many units; however, units cannot have more than one tenant each.
- There are also managers that manage the accounts and take payments for the account balances. The manager has a name, address, date of birth, and an employee ID. There can be many managers, however, there needs to be at least one manager because someone must always be managing an account.
- The facility has a name, address, facility ID, and phone number. The manager is employed by the facility and the storage units are owned by the facility.
- The main user of this database will be the facility managers and upper-level management. Tenants do not have or need access to the database as it contains sensitive information of other tenants and the account balance of each tenant.

ER Diagram

The figure below shows the ER diagram of the Storage Facility Database (SELFSTORAGEDB)

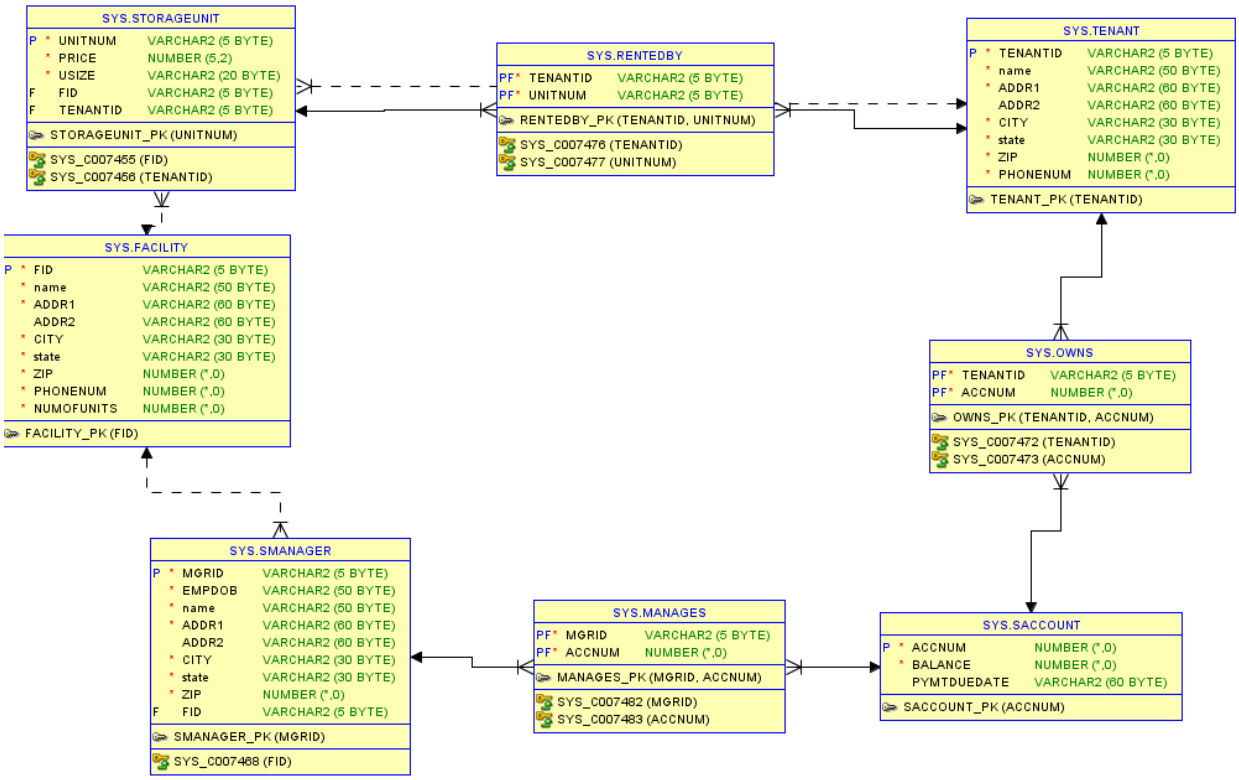


ER Diagram Uncaptured Constraints:

The following is a list of constraints that are not captured by the ER diagram of SELFSTORAGEDB

- The price of a Storage unit must be greater than zero.
- Account must have a balance that is greater than or equal to zero.
- Payment due date is an integer representing the day of the month that payment is due.

Relational Schema:



Relational Schema with Referential Integrity:

facility (<u>FID</u>, name, street, city, state, zip, phoneNum, numOfUnits)
storageUnit (<u>unitNum</u>, price, size)
    foreign key (FID) references facility (FID)
    foreign key (tenantID) references tenant (tenantID)
account (<u>accNum</u>, balance, pymtDueDate)
tenant (<u>tenantID</u>, name, zip, state, street, city, phoneNum)
manager (<u>mgrID</u>, empDOB, name, street, city, state, zip)
    foreign key (FID) references facility (FID)
owns (tenantID, accNum)

foreign key (tenantID) references tenant (tenantID)
foreign key (accNum) references account (accNum)
rentedBy (tenantID, unitNum)
foreign key (tenantID) references tenant (tenantID)
foreign key (unitNum) references storageUnit (unitNum)
manages (mgrID, accNum)
foreign key (mgrID) references manager(mgrID)
foreign key (accNum) references account(accNum)

Relational Table Details:

The relational schema given in the Relational Schema with Referential Integrity was mapped into the following tables in the SELFSTORAGEDB database. Primary keys have been underlined. Tables that have multiple attributes underlined represent composite keys.

Table Name	Attribute	Description
StorageUnit	unitNum	unique storage unit number
	price	positive decimal to show the price of a unit
	size	Size of a storage unit
	FID	the facility ID of the storage unit
	tenantID	The tenant's ID that owns the storage unit
Tenant	tenantID	unique tenant ID
	Name	tenant's name
	Zip	positieve integer showing the zip of the tenant
	State	the state that the tenant lives in
	Street	the street that the tenant lives on
	City	the city that the tenant lives in
	phoneNum	the phone number of the tenant
sAccount	accNum	a number linked to an account

	balance	the amount of money that is on the account
	pymtDueDate	the date that a payment is due
sManager	mgrID	unique manager ID
	empDOB	the manager's date of birth
	name	the manager's name
	Street	the street that the manager lives on
	City	the city that the manager lives in
	State	the state that the manager lives in
	zip	positieve integer showing the zip of the manager
	FID	the ID of the facility that the manager works for
Facility	FID	unique facility ID
	name	the facility's name
	street	the street that the facility is on
	city	the city that the facility is in
	state	the state that the facility is in
	Zip	positieve integer showing the zip of the facility
	phoneNum	the phone number of the facility
	numOfUnits	the number of units in the facility

owns	tenantID accNum	unique tenant ID number linked to account
rentedBy	tenantID unitNum	unique tenant ID unique storage unit number
manages	mgrID accNum	unique manager ID number linked to account

Queries:

Query Name	Description	Relations Accessed
unitsOfTenant	Retrieves information regarding all the units that a specific tenant is occupying, the number of each unit, the size of each unit, the price of each unit, and the ID of the tenant, grouped by the ID's of the tenant, and descending order from largest amount of units that a tenant is occupying to smallest. It will also display a total price from all the storage units a tenant occupies.	● storageUnit ● Tenant ● rentedBy
vacancyRatio	Retrieves a count of how many units are currently rented at each facility and the number of units in each facility. This information is used to calculate the percentage of units rented and sorted in descending order.	● storageUnit ● Facility ● ownedBy
tenantTotalBalance	Retrieves accounts which have a balance of less than zero, and displays the account balance, tenant name, and ID. These will be sorted in descending order, with the most negative balances being displayed first, grouped by tenant name. Will return the sum of balances along with the count of how many balances there are.	● tenant ● saccount ● owns
avgUnitsAndRevenue	Retrieves the number of units that each facility is currently using, averages the number of units per facility as well as the revenue each facility is bringing in every month.	● facility ● storageUnit

DDL

The following is an SQL definition of the tables for the SELFSTORAGEDB Database.

```
create table facility
(FID varchar(5) primary key,
 "name" varchar(50) not null,
 addr1 varchar(60) not null,
 addr2 varchar(60),
 city varchar(30) not null,
 "state" varchar(30) not null,
 zip integer not null check (zip > 0),
 phoneNum integer not null check (phoneNum > 0),
 numOfUnits integer not null check (numOfUnits > 0));

create table tenant
(tenantID varchar(5) primary key,
 "name" varchar(50) not null,
 addr1 varchar(60) not null,
 addr2 varchar(60),
 city varchar(30) not null,
 "state" varchar(30) not null,
 zip integer not null check (zip > 0),
 phoneNum integer not null check (phoneNum > 0));

create table storageUnit
(unitNum varchar(5) primary key,
 price decimal(5,2) not null check (price > 0),
 usize varchar(20) not null,
 FID varchar(5),
 tenantID varchar(5),
 foreign key (FID) references facility(FID),
 foreign key (tenantID) references tenant(tenantID));

create table saccount
(accNum integer primary key,
 balance integer not null check (balance > 0),
 pymtDueDate varchar(60));

create table smanager
(mgrID varchar(5) primary key,
```

```
 empDOB varchar(50) not null,
 "name" varchar(50) not null,
 addr1 varchar(60) not null,
 addr2 varchar(60),
 city varchar(30) not null,
 "state" varchar(30) not null,
 zip integer not null check (zip > 0),
 FID varchar(5),
 foreign key (FID) references facility(FID));


--relational tables
create table owns
(tenantID varchar(5) not null,
 accNum integer not null,
 primary key (tenantID,accNum),
 foreign key (tenantID) references tenant(tenantID),
 foreign key (accNum) references saccount(accNum));

create table rentedBy
(tenantID varchar(5),
 unitNum varchar(5) not null,
 primary key (tenantID,unitNum),
 foreign key (tenantID) references tenant(tenantID),
 foreign key (unitNum) references storageUnit(unitNum));

create table manages
(mgrID varchar(5) not null,
 accNum integer not null check (accNum ⩾ 0),
 primary key (mgrID,accNum),
 foreign key (mgrID) references smanager(mgrID),
 foreign key (accNum) references saccount(accNum));
```

Inserts & Queries

```
--facility
INSERT INTO
facility(FID,"name",addr1,addr2,city,"state",zip,phoneNum,numOfUnits)
            VALUES('00001','Facility','123 Facility
Dr',null,'Glendale','AZ','85000','1112223333','20');
INSERT INTO
facility(FID,"name",addr1,addr2,city,"state",zip,phoneNum,numOfUnits)
            VALUES('00002','Facility','456 Facility
Dr',null,'Phoenix','AZ','85020','6022223333','25');
```

```
 --tenant
 INSERT INTO tenant(tenantID, "name", addr1, addr2, city, "state", zip,
phoneNum)
            VALUES ('10001','Name1','123 Tenant
Ave',null,'Phoenix','AZ','85020','4801230456');
 INSERT INTO tenant(tenantID, "name", addr1, addr2, city, "state", zip,
phoneNum)
            VALUES ('10002','Name2','124 Tenant
Ave',null,'Phoenix','AZ','85012','4801230457');
 INSERT INTO tenant(tenantID, "name", addr1, addr2, city, "state", zip,
phoneNum)
            VALUES ('10003','Name3','125 Tenant
Ave',null,'Phoenix','AZ','85023','4801230458');
 INSERT INTO tenant(tenantID, "name", addr1, addr2, city, "state", zip,
phoneNum)
            VALUES ('10004','Name4','126 Tenant
Ave',null,'Scotsdale','AZ','85210','4801230459');
 INSERT INTO tenant(tenantID, "name", addr1, addr2, city, "state", zip,
phoneNum)
            VALUES ('10005','Name5','127 Tenant
Ave',null,'Scotsdale','AZ','85260','4801230460');

--storageUnit
INSERT INTO storageUnit(unitNum,price,usize,FID,tenantID)
            VALUES('00001','200.00','10×5','00001',null);
INSERT INTO storageUnit(unitNum,price,usize,FID,tenantID)
            VALUES('00002','200.00','10×10','00001',null);
INSERT INTO storageUnit(unitNum,price,usize,FID,tenantID)
            VALUES('00003','200.00','10×15','00001',null);
INSERT INTO storageUnit(unitNum,price,usize,FID,tenantID)
            VALUES('00004','200.00','10×20','00001',null);
INSERT INTO storageUnit(unitNum,price,usize,FID,tenantID)
            VALUES('00005','200.00','10×20','00001',null);
INSERT INTO storageUnit(unitNum,price,usize,FID,tenantID)
            VALUES('00006','200.00','10×20','00001',null);
INSERT INTO storageUnit(unitNum,price,usize,FID,tenantID)
            VALUES('00007','200.00','10×20','00001',null);
INSERT INTO storageUnit(unitNum,price,usize,FID,tenantID)
            VALUES('00008','200.00','10×20','00001',null);
INSERT INTO storageUnit(unitNum,price,usize,FID,tenantID)
            VALUES('00009','200.00','10×20','00001',null);
INSERT INTO storageUnit(unitNum,price,usize,FID,tenantID)
            VALUES('00010','200.00','10×20','00001',null);

--saccount
INSERT INTO saccount(accNum,balance,pymtDueDate)
            VALUES('001','100','11/03/2021');
INSERT INTO saccount(accNum,balance,pymtDueDate)
            VALUES('002','120','11/04/2021');
INSERT INTO saccount(accNum,balance,pymtDueDate)
            VALUES('003','90','11/02/2021');
INSERT INTO saccount(accNum,balance,pymtDueDate)
            VALUES('004','70','11/07/2021');
INSERT INTO saccount(accNum,balance,pymtDueDate)
```

```
            VALUES('005','200','11/04/2021');
INSERT INTO saccount(accNum,balance,pymtDueDate)
            VALUES('006','200','11/04/2021');
INSERT INTO saccount(accNum,balance,pymtDueDate)
            VALUES('007','300','11/05/2021');

--smanager

INSERT INTO smanager(mgrID,empDOB,"name",addr1,addr2,city,"state",zip,FID)
            VALUES('001','03/21/1999','name1','123 Manager
Dr',null,'Goodyear','AZ','85000','00001');
INSERT INTO smanager(mgrID,empDOB,"name",addr1,addr2,city,"state",zip,FID)
            VALUES('002','03/24/1976','name2','124 Manager
Dr',null,'Goodyear','AZ','85000','00001');
INSERT INTO smanager(mgrID,empDOB,"name",addr1,addr2,city,"state",zip,FID)
            VALUES('003','03/30/1985','name3','125 Manager
Dr',null,'Goodyear','AZ','85000','00001');

--owns
INSERT INTO owns(tenantID, accNum)
            VALUES('10001','001');
INSERT INTO owns(tenantID, accNum)
            VALUES('10002','002');
INSERT INTO owns(tenantID, accNum)
            VALUES('10003','003');
INSERT INTO owns(tenantID, accNum)
            VALUES('10004','004');
INSERT INTO owns(tenantID, accNum)
            VALUES('10005','005');
INSERT INTO owns(tenantID, accNum)
            VALUES('10001','006');
INSERT INTO owns(tenantID, accNum)
            VALUES('10002','007');
--rentedBy
INSERT INTO rentedBy(tenantID,unitNum)
            VALUES('10001','00001');
INSERT INTO rentedBy(tenantID,unitNum)
            VALUES('10002','00002');
INSERT INTO rentedBy(tenantID,unitNum)
            VALUES('10003','00003');
INSERT INTO rentedBy(tenantID,unitNum)
            VALUES('10004','00004');
INSERT INTO rentedBy(tenantID,unitNum)
            VALUES('10005','00005');
INSERT INTO rentedBy(tenantID,unitNum)
            VALUES('10001','00006');
INSERT INTO rentedBy(tenantID,unitNum)
            VALUES('10001','00007');
INSERT INTO rentedBy(tenantID,unitNum)
            VALUES('10002','00008');
INSERT INTO rentedBy(tenantID,unitNum)
            VALUES('10004','00009');
INSERT INTO rentedBy(tenantID,unitNum)
            VALUES('10005','00010');
```

```
-- Query: vacancyRatio -- Steven Anderson
SELECT storageUnit.fid, COUNT(*) as Rented, numOfUnits,
       100 - ROUND(COUNT(*) * 100 / numOfUnits,1) as Vacancy
FROM storageUnit
INNER JOIN Facility ON storageUnit.fid=facility.fid
GROUP BY storageUnit.fid, numOfUnits
ORDER BY Vacancy DESC;
```

FID	RENTED	NUMOFUNITS	VACANCY
00001	10	20	50

```
-- Query: tenantTotalBalance -- Adam Keech
select tenant."name" as "Name", tenant.tenantID as tID, sum(balance) as
total_Balance, count(balance) as num_Balances
from tenant, owns, saccount
where tenant.tenantID = owns.tenantID and saccount.accNum = owns.accNum
group by tenant."name", tenant.tenantID
order by total_Balance DESC;
```

Name	TID	TOTAL_BALANCE	NUM_BALANCES
Name2	10002	420	2
Name1	10001	300	2
Name5	10005	200	1
Name3	10003	90	1
Name4	10004	70	1

```
-- Query: unitsOfTenant -- Cory Gregory
SELECT S.unitNum, S.usize, S.price, R.tenantID, sum(S.price), count(S.usize)
FROM tenant T, storageUnit S, rentedBy R
WHERE T.tenantID=R.tenantID and S.unitNum=R.unitNum
GROUP BY R.tenantID, S.unitNum, S.usize, S.price
ORDER BY count(S.usize) DESC;
```

UNITN	USIZE	PRICE	TENAN	SUM(S.PRICE)	COUNT(S.USIZE)
00001	10x5	200	10001	200	1
00006	10x20	200	10001	200	1
00007	10x20	200	10001	200	1
00002	10x10	200	10002	200	1
00010	10x20	200	10005	200	1
00003	10x15	200	10003	200	1
00004	10x20	200	10004	200	1
00009	10x20	200	10004	200	1
00005	10x20	200	10005	200	1
00008	10x20	200	10002	200	1

```
-- Query: avgUnitsAndRevenue-- Scott Rowe

--Finds the average number of storage units at each facility
avgNumUnitsAtFacility :=
SELECT AVG(numOfUnits) AS avgNumUnits
FROM facility;

--Finds the sum of the Revenue of all units
totalRevenueFromUnits :=
SELECT SUM(price) AS totalRevenue
FROM storageUnit;

--Divides the sum of the revenue for every facility in order to find the
average revenue per facility
avgRevenuePerFacility :=
SELECT totalRevenue / COUNT(facility) AS avgRevenue
FROM totalRevenueFromUnits, facility;

--Combines and organizes the averages
avgUnitsAndRevenue :=
SELECT avgNumUnits, avgRevenue
FROM avgNumUnitsAtFacility, avgRevenuePerFacility
GROUP BY avgRevenuePerFacility.avgRevenue, avgNumUnitsAtFacility.avgNumUnits
ORDER BY avgRevenue DESC;
```

Group Status Report:

GROUP #: 8 GROUP NAME: PHASE #: 3

Overview of progress on project as of November 23:

Final adjustments were made in the database and example data has been added to demonstrate queries.

CONTRIBUTIONS OF GROUP MEMBERS:

Phase Leader: Corey Gregory

- QUERY IMPLEMENTATION: unitsOfTenant

Phase Recorder: Scott Rowe

- QUERY IMPLEMENTATION: avgUnitsAndRevenue

Phase Checker: Adam Keech

- QUERY IMPLEMENTATION: tenantTotalBalance
- Revised Phase 3 deliverable.
- Added relational tables to DB and created final DDL.

Technical Advisor: Steven Anderson

- QUERY IMPLEMENTATION: vacancyRatio
- Revised Phase 3 deliverable.
- Group status report