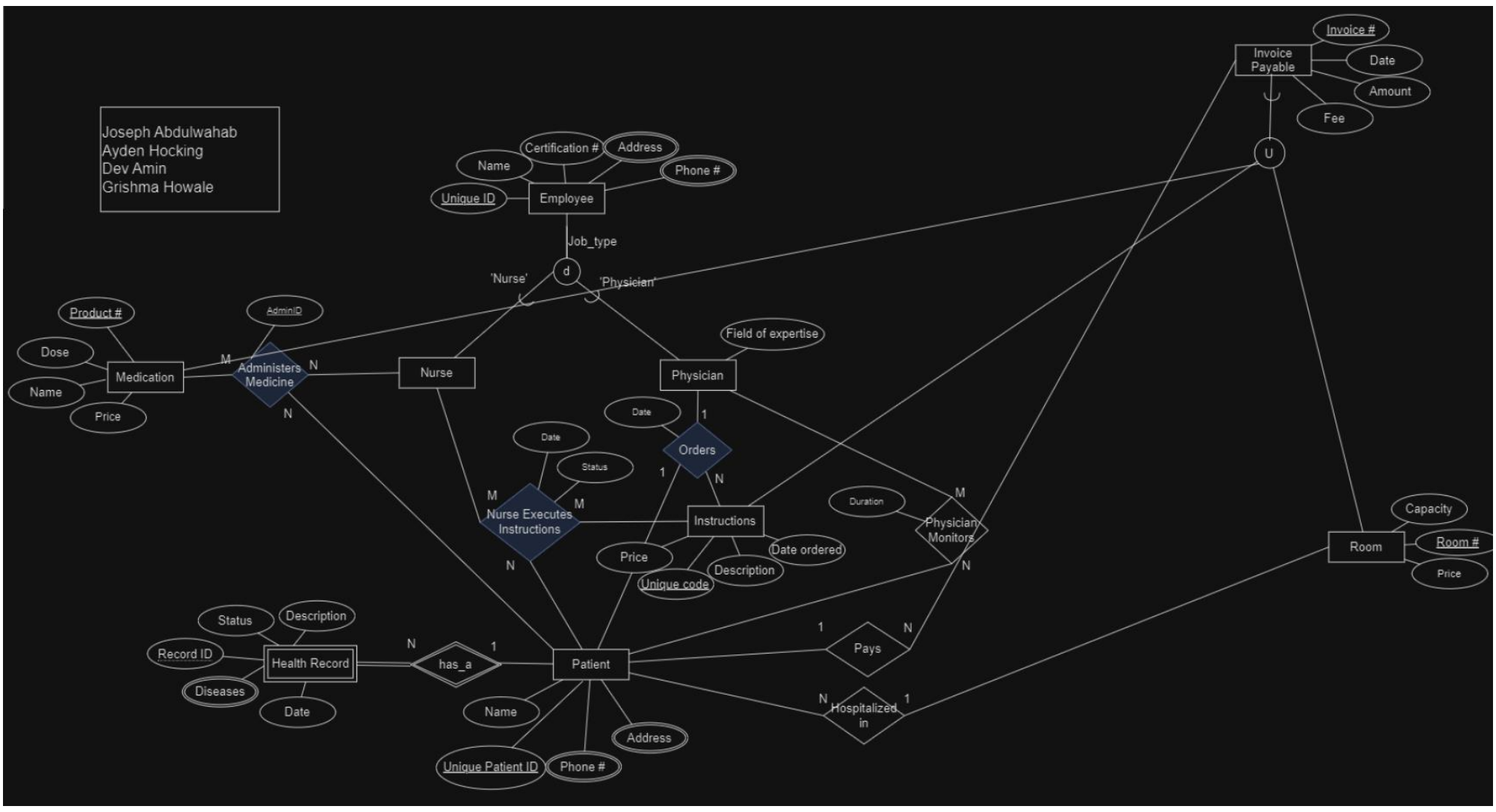
Joseph Abdulwahab, Ayden Hocking, Grishma Howale, Dev Amin

Assumptions, (E)ERD, Relations and keys



Assumptions:

- Added an attribute for medicine price for any in-house given medicine which can be referenced in invoice payable.
- Created Relations, shaded in green, for multivariable attributes.

Set of relations:

Employee: (UniqueID, Name, CertificationNum)
Primary key: {UniqueID}
EmployeePhoneNums(EmpID, Phone)
Primary key: {EmpID, Phone}
Foreign key: {EmpID references Employee(UniqueID)}
EmployeeAddresses(EmpID, Address)
Primary key: {EmpID, Address}
Foreign key: {EmpID references Employee(UniqueID)}

Nurse(NurNum)
Primary key: {NurNum}
Foreign key: {NurNum references Employee(UniqueID)}
Physician(PhyNum, FieldOfExperience)
Primary key: {PhyNum}
Foreign key: {PhyNum references Employee(UniqueID)}

Room(RoomNum, Capacity, Price)
Primary key: {RoomNum}

Patient(PatID, PhysAssigned, RoomAssigned, Name)
Primary key: {PatID}
Foreign key: {PhysAssigned references Physician(PhyNum), RoomAssigned references Room(RoomNum)}
PatientPhoneNums(PID, Phone)
Primary key: {PID, Phone}
Foreign key: {PID references Patient(PatID)}
PatientAddresses(PID, Address)
Primary key: {PID, Address}
Foreign key: {PID references Patient(PatID)}

Instructions(UniqueID, PhysPrescribed, Price, Description, Date_ordered)
Primary key: {UniqueID}
Foreign key: {PhysPrescribed references Physician(PhyNum)}

Orders(PhysicianID, PatientID, InstructionID , Date)
Primary key: {PhysicianID, PatientID}
Foreign key: {PhysicianID references Physician(PhyNum),
PatientID references Patient(PatID),
InstructionID references Instructions(UniqueID)}

Medication(ProductNum, Dose, Name, Price)
Primary key: {ProductNum}

Administers_Medicine(AdminID, NurseID, MedicineID, PatientGiven)
Primary key: {AdminID, NurseID, ProductID, PatientGiven}
Foreign key: {NurseID references Nurse(NurNum),
MedicineID references Medication(ProductNum),
PatientGiven references Patient(PatID)}

Health_Record(RecordID, PatientID, Status, Date, Description)
Primary key: {RecordID, PatientID}
Foreign key: {PatientID references Patient(PatID)}
Diseases(PID, Disease)
Primary key: {PID, Disease}
Foreign key: {PID references Health_Record(PatientID)}

Invoice_Payable(InvoiceNum, PatientTreated, RoomOccupied, InstructionsPerformed, MedicineGiven, Date, Amount, Fee)
Primary key: {InvoiceNum}
Foreign key: {PatientTreated references Patient(PatID),
RoomOccupied references Room(RoomNum),
InstructionsPerformed references Instructions(UniqueID),
MedicineGiven references Administers_Medicine(AdminID) }

Nurse_Executes_Instructions(NurseID, InstructionID, PatientID, Date, Status)
Primary key: {NurseID, InstructionID, PatientID}
Foreign key: {NurseID references Nurse(NurNum), InstructionID references Instructions(UniqueID),
PatientID references Patient(PatID)}

Physician_Moniors(PhysicianID, PatientID, DurationInHours)
Primary key: {PhysicianID, PatientID}
Foreign key: {PhysicianID references Physician(PhyNum), PatientID references Patient(PatID)}

Joseph Abdulwahab, Ayden Hocking, Grishma Howale, Dev Amin

3 View Queries

1) This view shoes each patient's assigned physician and room number

```
Drop view if exists assignment;
create view assignment as
select patient, Employee.name as physician, RoomNum
from Employee, (select Patient.name as patient, Physician.PhyNum as phyid, RoomNum
                from Patient, Physician, Room
                where Patient.PhysAssigned = Physician.PhyNum and Patient.RoomAssigned = Room.RoomNum) as base
where Employee.UniqueID = base.phyid;
select * from assignment;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

patient	physician	RoomNum
Mikey Greene	Sandy Rivera	F54
Mariam Verde	Michael Angelo	A20
Dolly Linda	Joseph Abdulwahab	C50
Gerald quan	Jacob Carter	A23
Linda olive	Harrison Ford	B23

---

2) This view shows which nurse gave what medicine to which patient

```
Drop view if exists medicineGiven;
create view medicineGiven as
select eventID, Employee.name as nurse, Medication.name, Medication.dose as dose, Patient.Name as patient
from Employee, Medication, Patient,
    (select am.AdminID as eventID, am.NurseID as nID, am.MedicineID as mID, am.PatientGiven as pID
    from Administers_Medicine as am) as base
where base.nID = Employee.UniqueID and Medication.ProductNum = base.mID and Patient.PatID = base.pID;
select * from medicineGiven;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

eventID	nurse	name	dose	patient
493034CEWC	Selena Hernandez	Melatonin	75.43	Mikey Greene
34r43FFRfc	Jerry Arnold	Vicidin	5.634	Mariam Verde
ERFIREF343	Kelly Phung	Codeine	4.34	Linda olive
RTRIF934T5	Damian Rock	Aspirin	2.8	Gerald quan
LTIFYIF65Y	Sarah Hartlock	Tylenol	12.43	Dolly Linda

---

3) This view is the composition of the invoice payable. The fee is 20% of the instructions cost

```
Drop view if exists invoice;
create view invoice as
select InvoiceID, date, patient, description, roomOccupied, roomPrice, Medication.name as MedicineAdministered, price as MedicineCost, Instructionsamount,
       ROUND(0.20*base.Instructionsamount, 2) as fees, ROUND((roomPrice+price+Instructionsamount+(0.20*base.Instructionsamount)), 2) as Total_Payable_Due
from Medication join
    (select invoiceNum as InvoiceID, p.name as patient, inst.Description, Room.RoomNum as roomOccupied,
    Room.Price as roomPrice, am.MedicineID as mID, date, inst.Price as Instructionsamount
    from Invoice_payable as ip
        join Patient p on ip.PatientTreated = p.PatID
        join Room on ip.RoomOccupied = Room.RoomNum
        join Instructions inst on ip.InstructionsPerformed = inst.UniqueID
        join Administers_Medicine as am on ip.MedicineGiven = am.AdminID) as base
on Medication.ProductNum = base.mID;
select * from invoice;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

InvoiceID	date	patient	description	roomOccupied	roomPrice	MedicineAdministered	MedicineCost	Instructionsamount	fees	Total_Payable_Due
739457393	2020-05-25	Mikey Greene	Cancer Screening	F54	12.9	Melatonin	22.09	2340.65	468.13	2843.77
573957383	2019-04-23	Mariam Verde	Diagnosis	A20	34.54	Vicidin	43.25	250	50	377.79
284835374	2019-08-15	Linda olive	CAT Scan	B23	1400.43	Codeine	65.87	8090.23	1618.05	11174.58
943050393	2020-04-23	Gerald quan	Skin Graph	A23	2303.34	Aspirin	10.54	12056.93	2411.39	16782.2
758350987	2019-02-25	Dolly Linda	Lung Transplant	C50	504.23	Tylenol	5.43	5004.45	1000.89	6515

3 Join Queries

1) What is the Patient's room number?

```
select name, Room.RoomNum
from Patient join Room on Patient.RoomAssigned = Room.RoomNum;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

name	RoomNum
Mikey Greene	F54
Mariam Verde	A20
Dolly Linda	C50
Gerald quan	A23
Linda olive	B23

---

2) How long has the patient been monitored by a physician?

```
select Employee.name as Physician, Patient.name as Patient, DurationInHours
from Physician_Moniters as py join Patient on py.PatientID = Patient.PatID
join Employee on py.PhysicianID = Employee.UniqueID;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Physician	Patient	DurationInHours
Sandy Rivera	Mikey Greene	4.34
Michael Angelo	Mariam Verde	54.3
Michael Angelo	Dolly Linda	12.43
Joseph Abdulwahab	Dolly Linda	23.2
Jacob Carter	Gerald quan	12.3
Sandy Rivera	Gerald quan	43.3
Harrison Ford	Linda olive	4.56

---

3) What is each Patient's disease(s)?

```
select name as patient, Disease
from Diseases join Patient on Diseases.PID = Patient.PatID;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

patient	Disease
Mikey Greene	Covid
Mikey Greene	STD
Mariam Verde	Flu
Mariam Verde	Zika
Dolly Linda	Acne
Gerald quan	Sickle Cell
Linda olive	Malaria

Joseph Abdulwahab, Ayden Hocking, Grishma Howale, Dev Amin

4 Aggregation Queries

1) What is the total amount of hospital payables?

```
52 •  select SUM(Total_Payable_Due) as Total_payables
53    from invoice; -- from view created previously
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Total_payables
37693.34

---

2) How many patients received medication over $20?

```
55 •  select COUNT(*)
56    from Medication
57    where Medication.Price > 20;
```

Result Grid | Filter Rows: | Export:

COUNT(*)
3

---

3) What is the average price spent on rooms with capacity under 5?

```
59 •  select ROUND(AVG(Price), 2) as AvgPrice_CapUnder5
60    from Room
61    where capacity < 5;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

AvgPrice_CapUnder5
1402.67

---

4) Show which physicians monitored more than 1 patient.

```
63 •  select Name as Physician, base.PatientsMonitored
64    from Employee join (
65    select PhysicianID, Count(PatientID) as PatientsMonitored
66    from Physician_Moniters
67    group by PhysicianID
68    having count(PatientID) > 1) as base on UniqueID = PhysicianID;
69
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Physician	PatientsMonitored
Michael Angelo	2
Sandy Rivera	2

3 Nested Queries

1) Return every Patient that paid Instructions that cost over $5,000

```
72 •  select
73        (select name from Patient where PatID = Orders.PatientID) as PatientNames,
74        (select price from Instructions where Instructions.UniqueID = Orders.InstructionID) as Price
75    from Orders
76    where Orders.InstructionID in (
77        select UniqueID
78        from Instructions
79        where price > 5000
80    );
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

PatientNames	Price
Linda olive	8090.23
Gerald quan	12056.93
Dolly Linda	5004.45

---

2) Return Patient(s) that were monitored by Physician Joseph Abdulwahab

```
82 •  select name as patient
83    from Patient
84    where PatID in (
85        select PatientID
86        from Physician_Moniters
87        where PhysicianID in (
88            select PhyNum
89            from Physician
90            where PhyNum in (
91                select UniqueID
92                from Employee
93                where Name like "Joseph Abdulwahab"
94                )
95            )
96        );
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

patient
Dolly Linda

---

3) What patient(s) were monitored for over 10 hours by physician and how many hours were they monitored?

```
98 •  select Employee.name as Physician, Patient.name as Patient, base.DurationInHours
99    from Employee join
100   (select PhysicianID, PatientID, DurationInHours
101   from Physician_Moniters
102   where DurationInHours > 10) as base on Employee.UniqueID = PhysicianID
103   join Patient on Patient.PatID = base.PatientID;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Physician	Patient	DurationInHours
Michael Angelo	Mariam Verde	54.3
Michael Angelo	Dolly Linda	12.43
Jacob Carter	Gerald quan	12.3
Joseph Abdulwahab	Dolly Linda	23.2
Sandy Rivera	Gerald quan	43.3

Joseph Abdulwahab, Ayden Hocking, Grishma Howale, Dev Amin

3 Triggers

1) Insert a patient health record once the patient is admitted to the hospital today

```
107 •  DROP TRIGGER IF EXISTS insert_patient_record;
108    DELIMITER //
109 •  CREATE TRIGGER insert_patient_record
110    AFTER INSERT ON patient
111    FOR EACH ROW BEGIN
112        INSERT INTO Health_Record (PatientID, Status, Date, description)
113        VALUES (NEW.PatID, 'Admitted', Now(), 'New patient record created');
114    END; //
115    DELIMITER ;
116 •  select * from Health_Record; -- do an insert statement after the trigger to see the effects.
117                                 -- Ideally, run schema, trigger, then insert data, then finally view the relation.
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

RecordID	PatientID	Status	Date	Description
1	678564320	Admitted	2024-04-27	New patient record created
2	768493756	Admitted	2024-04-27	New patient record created
3	820483754	Admitted	2024-04-27	New patient record created
4	340384573	Admitted	2024-04-27	New patient record created
5	204387303	Admitted	2024-04-27	New patient record created

---

2) After records are inserted in the orders relation today, update instructions to have today's date_ordered

Before trigger:

```
105 •  select * from instructions;
```

Result Grid | Filter Rows: | Edit: | Export/I

UniqueID	PhysPrescribed	Price	Description	Date_ordered
203404345	904875738	2340.65	Cancer Screening	2020-04-23
343480034	382539503	8090.23	CAT Scan	2019-08-11
384954945	758603759	12056.93	Skin Graph	2020-02-12
503494003	340357359	250	Diagnosis	2019-02-13
758409857	847590547	5004.45	Lung Transplant	2019-02-23
NULL	NULL	NULL	NULL	NULL

After trigger:

```
120 •  DROP TRIGGER IF EXISTS Modify_date;
121    DELIMITER //
122 •  CREATE TRIGGER Modify_date
123    AFTER INSERT ON Orders
124    FOR EACH ROW BEGIN
125        Update Instructions set Date_ordered = NOW()
126        Where Instructions.UniqueID = NEW.InstructionID; -- new InstructionID b/c this is after creating order records. cannot do Orders.InstructionID
127    END; //
128    DELIMITER ;
129 •  select * from Instructions;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

UniqueID	PhysPrescribed	Price	Description	Date_ordered
203404345	904875738	2340.65	Cancer Screening	2024-04-27
343480034	382539503	8090.23	CAT Scan	2024-04-27
384954945	758603759	12056.93	Skin Graph	2024-04-27
503494003	340357359	250	Diagnosis	2024-04-27
758409857	847590547	5004.45	Lung Transplant	2024-04-27

---

3) Update the room capacity by -1 each time a patient is admitted to the room.

Before trigger:

```
106 •  select * from room;
```

Result Grid | Filter Rows: | Edit

RoomNum	Capacity	Price
A20	12	34.54
A23	1	2303.34
B23	2	1400.43
C50	4	504.23
F54	43	12.9
NULL	NULL	NULL

After trigger:

```
132 •  DROP TRIGGER IF EXISTS updateRoomCapacity;
133    DELIMITER //
134 •  CREATE TRIGGER assign_nurse_to_instruction
135    AFTER INSERT ON Patient
136    FOR EACH ROW
137    BEGIN
138        UPDATE Room set Capacity = Capacity - 1
139        where New.RoomAssigned = RoomNum;
140    END;//
141    DELIMITER ;
142 •  select * from room;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Co

RoomNum	Capacity	Price
A20	11	34.54
A23	0	2303.34
B23	1	1400.43
C50	3	504.23
F54	42	12.9

Joseph Abdulwahab, Ayden Hocking, Grishma Howale, Dev Amin

3 Transactions

1) Add a record with commit

Before Transaction:

```
151 •   select * from employee;
```

Result Grid | Filter Rows: | Expor

UniqueID	Name	CertificationNum
230838583	Jerry Arnold	FVT54YT67U
255939579	Kelly Phung	FREQGT230Y
340357359	Michael Angelo	LPOKF65YU9
382539503	Harrison Ford	RCFYHSWRT
573957494	Sarah Hartlock	GYUFEQDFT6
690348583	Selena Hernandez	DEQAS54TY6
758603759	Jacob Carter	GYE55TYOUI
847590547	Joseph Abdulwahab	TUH4yh78T6
904875738	Sandy Rivera	U765TYFR5T
929548248	Damian Rock	JUYTF67TY5

After Transaction:

```
145 •   SET autocommit = 0;
146 •   START TRANSACTION;
147 •   INSERT INTO Employee VALUES (657420098, "Jake Okra", "GJYTY65&9");
148 •   COMMIT;
149     -- delete from employee where UniqueID = 657420098;
150 •   select * from employee;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

UniqueID	Name	CertificationNum
230838583	Jerry Arnold	FVT54YT67U
255939579	Kelly Phung	FREQGT230Y
340357359	Michael Angelo	LPOKF65YU9
382539503	Harrison Ford	RCFYHSWRT
573957494	Sarah Hartlock	GYUFEQDFT6
657420098	Jake Okra	GJYTY65&9
690348583	Selena Hernandez	DEQAS54TY6
758603759	Jacob Carter	GYE55TYOUI
847590547	Joseph Abdulwahab	TUH4yh78T6
904875738	Sandy Rivera	U765TYFR5T
929548248	Damian Rock	JUYTF67TY5

---

2) Rollback

Before Rollback:

```
153 •   START TRANSACTION;
154 •   INSERT INTO Room VALUES('T65', 8, 999.843);
155 •   select * from Room;
```

Result Grid | Filter Rows: | Export: | Wrap Cell C

RoomNum	Capacity	Price
A20	11	34.54
A23	0	2303.34
B23	1	1400.43
C50	3	504.23
F54	42	12.9
T65	8	999.843

After Rollback:

```
153 •   START TRANSACTION;
154 •   INSERT INTO Room VALUES('T65', 8, 999.843);
155 •   select * from Room;
156 •   ROLLBACK;
157 •   select * from Room;
```

Result Grid | Filter Rows: | Export: | Wrap Cell C

RoomNum	Capacity	Price
A20	12	34.54
A23	1	2303.34
B23	2	1400.43
C50	4	504.23
F54	43	12.9

---

3) All or nothing

```
DROP PROCEDURE IF EXISTS insert_all;
DELIMITER //
CREATE PROCEDURE insert_all()
BEGIN
    DECLARE rollback_ bool DEFAULT 0;          -- define and initialize the variable rollback_ to 0
    DECLARE CONTINUE HANDLER FOR SQLEXCEPTION  -- define a handler method that sets rollback_ to 1 whenver
    BEGIN                                      -- we face SQLEXCEPTION as a result of running the statements.
        SET @rollback_ = 1;
    END;

    START TRANSACTION;
    INSERT INTO Room VALUES('AAA', 8, 999.843);
    INSERT INTO Room VALUES('AAA', 9, 32.23); -- should not be added because breaks primary key

    IF @rollback_ THEN
        ROLLBACK;
        SELECT 'Error occccccured' as message;
    ELSE
        COMMIT;
        select 'committtted' as message;
    END IF;
END //
DELIMITER ;
```