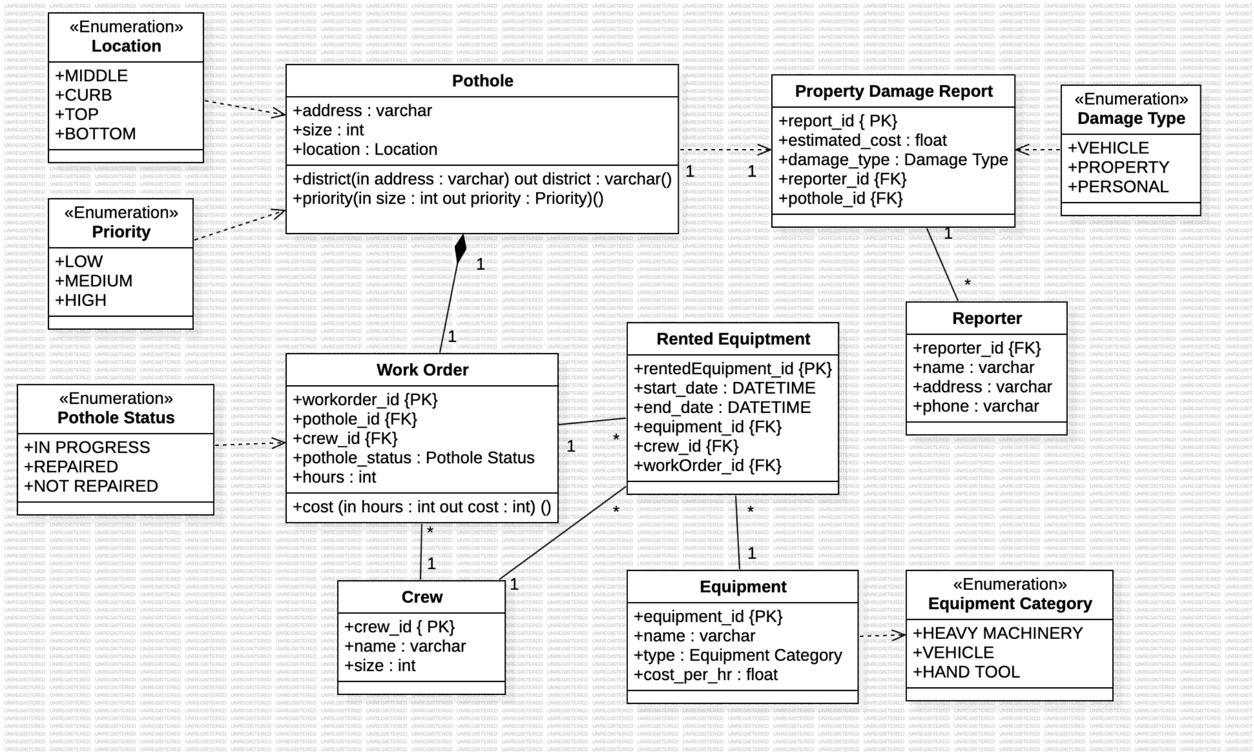
Project 3 Report

Tiana Thomas and John R. Smith

GitHub Repo: https://github.com/jrsmith03/cs347-p3

Database Design



We designed our schema with flexibility and expansion in mind. Take the following workflow as an example of how our schema works:

- A user wants to report a pothole, so they register with a user account. This creates an entry in the Reporter table, which can later be referenced during User Authentication.
- Now, the user fills out a report, creating an entry in the PropertyDamageReport table. Our schema could support the addition of a trigger that would automatically create a corresponding field in the Potholes table given the qualities described in the report - it currently stores a reference to a single pothole ID.
- After some city official presumably evaluates the user report, they will create a WorkOrder entry. Officials will have already created entries for Crews, so the official will simply assign a reference to an existing crew for the work order.
    - Equipment assignments are handled by creating an entry in the RentedEquiptment table.
    - Throughout the lifecycle of the work order, the application can notify the user of status updates (we designed a query that would facilitate this). Finally, once a pothole is repaired, the official can mark the status as Complete and this change

will be reflected across the work order. The application could use this schema to reveal further information in user reports, such as whether a crew as been assigned and the estimated timeframe for completion (we also have some example queries)

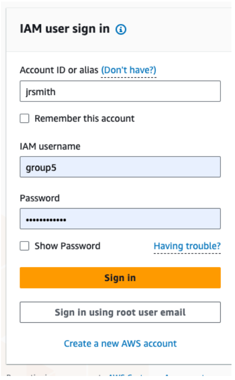
Most table design choices are evident in the UML diagram. If the database were to expand, our diagram featured function definitions for district and priority in Pothole and cost in WorkOrder.

- For district, the application may utilize a geography API that, given an address, can see the corresponding district from a municipal map. Note that this will need to be determined in something like a Python script because we'd have to use some external RESTful API library to do these queries. We'd format the output and write an INSERT query (using the same cursor that's reading the DB) to insert the data.
- For priority, the DDL may define a function that factors pothole size and district together and map it to some scale. For example, it may be that a pothole of size 3 in the middle of downtown or on a freeway will have a higher priority than an equivalently-sized pothole on some backroads.

We decided to create a junction table (RentedEquipment) because we could see the potential existence of a many to many relation between Crews and Equipment (i.e. many crews may rent out the same piece of equipment, and it may even be shared).

- A crew will be assigned some number of RentedEquipments, and each RentedEquipment will correspond to a given work order.

AWS and PostGres RDS Setup



IAM user sign in ⓘ

Account ID or alias (Don't have?)

jrsmith

□ Remember this account

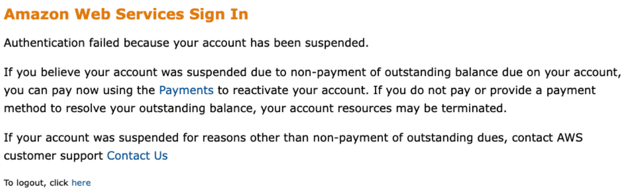
IAM username

group5

Password

••••••••••••

□ Show Password Having trouble?



Amazon Web Services Sign In

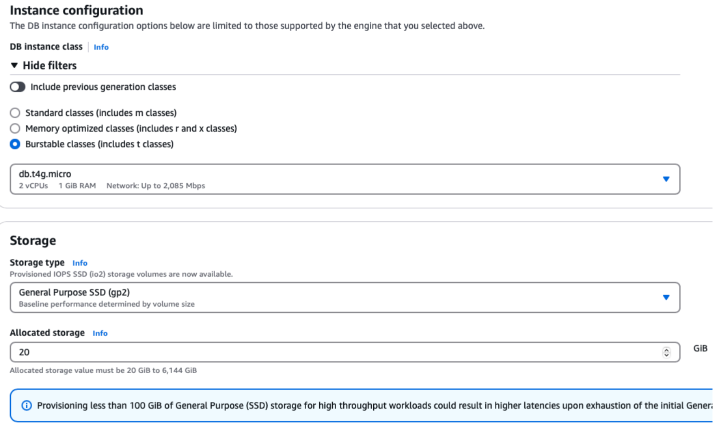
Authentication failed because your account has been suspended.

If you believe your account was suspended due to non-payment of outstanding balance due on your account, you can pay now using the Payments to reactivate your account. If you do not pay or provide a payment method to resolve your outstanding balance, your account resources may be terminated.

If your account was suspended for reasons other than non-payment of outstanding dues, contact AWS customer support Contact Us

To logout, click here

Creating the AWS account was a bit confusing for me because I had my email and some IAM credentials linked from a previous course and personal project. I had since closed that account, but AWS did not intuitively allow me to 're-open' it on the free tier. I ended up just creating an account with a separate email, which was trivial.



Instance configuration

The DB instance configuration options below are limited to those supported by the engine that you selected above.

DB instance class Info

▼ Hide filters

Include previous generation classes

○ Standard classes (includes m classes)

○ Memory optimized classes (includes r and x classes)

● Burstable classes (includes t classes)

db.t4g.micro
2 vCPUs 1 GiB RAM Network: Up to 2,085 Mbps

Storage

Storage type Info
Provisioned IOPS SSD (io2) storage volumes are now available.

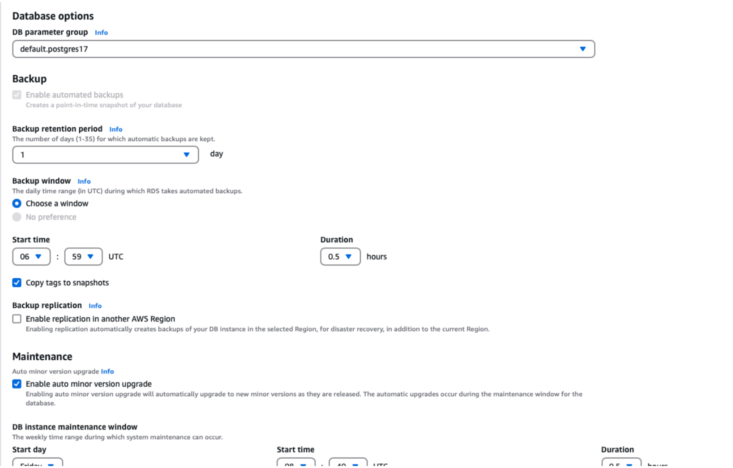
General Purpose SSD (gp2)
Baseline performance determined by volume size

Allocated storage Info

20 GiB

Allocated storage value must be 20 GiB to 6,144 GiB

ⓘ Provisioning less than 100 GiB of General Purpose (SSD) storage for high throughput workloads could result in higher latencies upon exhaustion of the initial Genera



Database options

DB parameter group Info

default.postgres17

Backup

☑ Enable automated backups
Creates a point-in-time snapshot of your database

Backup retention period Info
The number of days (1-35) for which automatic backups are kept.

1 day

Backup window Info
The daily time range (in UTC) during which RDS takes automated backups.

● Choose a window

○ No preference

Start time 06 : 59 UTC Duration 0.5 hours

☑ Copy tags to snapshots

Backup replication Info

□ Enable replication in another AWS Region
Enabling replication automatically creates backups of your DB instance in the selected Region, for disaster recovery, in addition to the current Region.

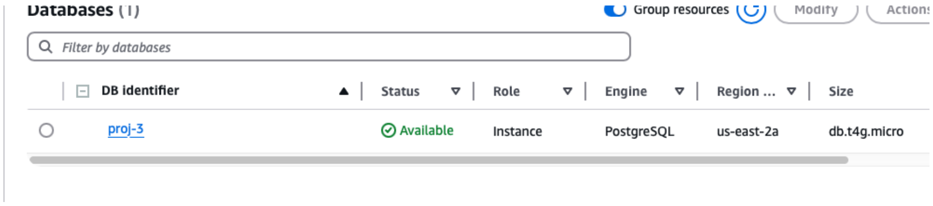
Maintenance

Auto minor version upgrade Info

☑ Enable auto minor version upgrade
Enabling auto minor version upgrade will automatically upgrade to new minor versions as they are released. The automatic upgrades occur during the maintenance window for the database.

DB instance maintenance window
The weekly time range during which system maintenance can occur.

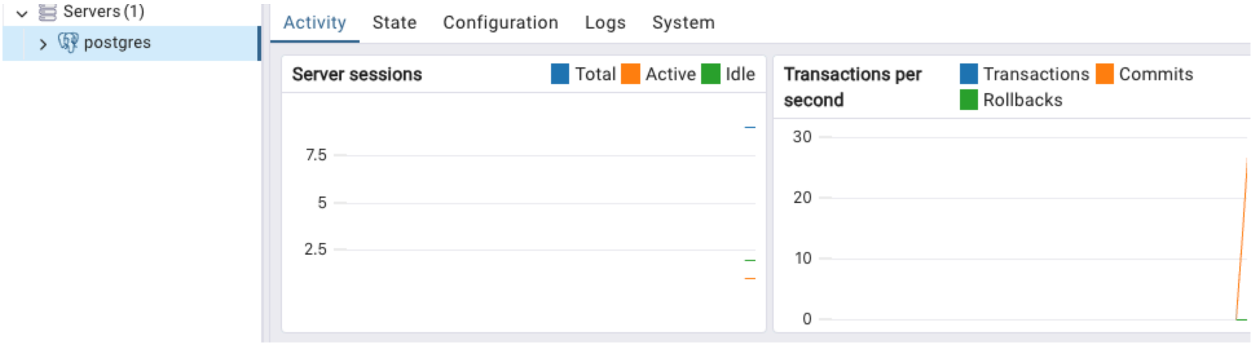
Start day Start time Duration



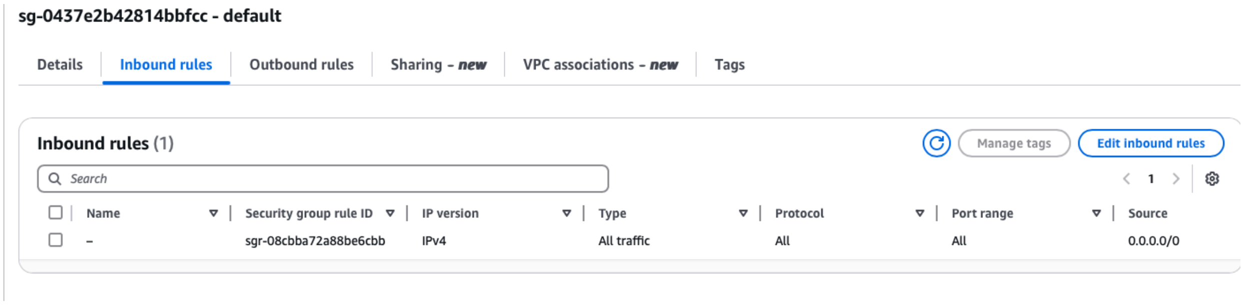
Databases (1)

DB identifier	Status	Role	Engine	Region ...	Size
proj-3	Available	Instance	PostgreSQL	us-east-2a	db.t4g.micro

Creating the PostGres RDS instance was quite straightforward. We followed the linked guide and made minimal changes to the configuration given the very low resource demand of this small database. It was interesting seeing all of the options and configuration for replication, backup, and compute resources however.



I installed pgAdmin to provide a simple interface into the DB. I was able to figure out what info needed to be copied over from AWS in order to provide authentication. However, I ran into a snag because the default security rule only allowed connections from other EC2 instances.

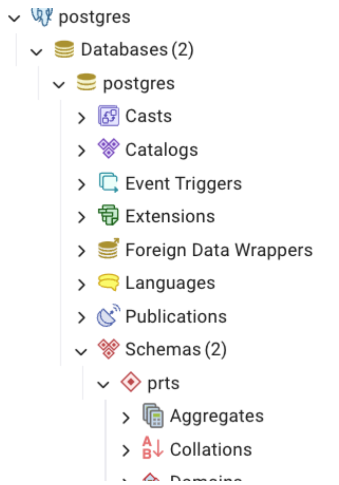


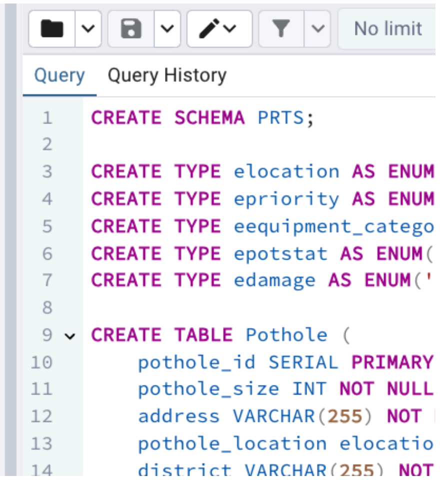
sg-0437e2b42814bbfcc - default

Inbound rules (1)

Name	Security group rule ID	IP version	Type	Protocol	Port range	Source
–	sgr-08cbba72a88be6cbb	IPv4	All traffic	All	All	0.0.0.0/0

Modifying the security group was confusing, but the Medium article was a big help. I got a bit lost about whether I was creating a new group or a new inbound rule. Once I allowed connections from any IPV4 port range, however, I was able to get in easily using pgAdmin. Do note that we obviously wouldn't want this open of a database in a real production environment; for this tiny project, however, it made sense to just give basic authentication via a password.



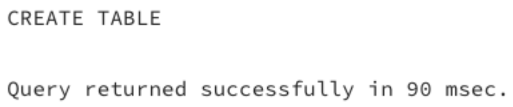


```
CREATE SCHEMA PRTS;

CREATE TYPE elocation AS ENUM
CREATE TYPE epriority AS ENUM
CREATE TYPE eequipment_catego
CREATE TYPE epotstat AS ENUM(
CREATE TYPE edamage AS ENUM('

CREATE TABLE Pothole (
    pothole_id SERIAL PRIMARY
    pothole_size INT NOT NULL
    address VARCHAR(255) NOT
    pothole_location elocatio
    district VARCHAR(255) NOT
```

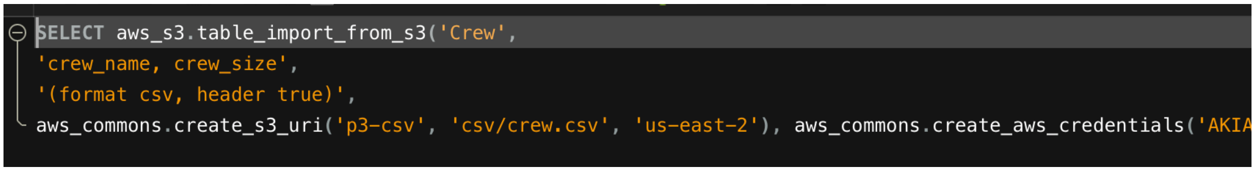
Once we were authenticated, we were able to easily copy over our DDL file and create the schema. This worked very similarly to how it did with MySQL and in Workbench.



```
CREATE TABLE

Query returned successfully in 90 msec.
```

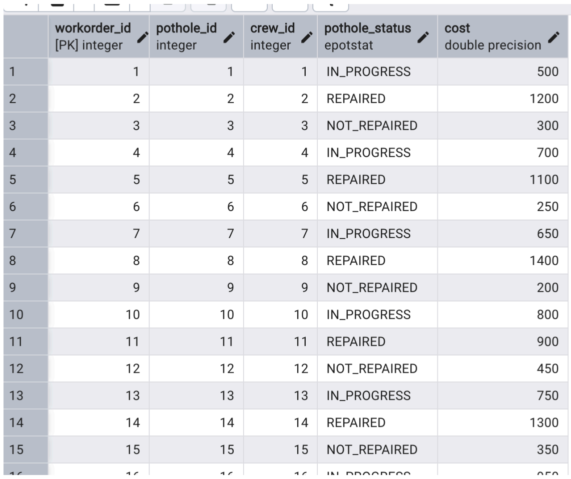
By far the most difficult part of this project was importing the data. Following Amazon's guide, we first built a small S3 and uploaded our CSVs. What was difficult was giving our RDS knowledge of the location of this data (and, of course, permission to access it). We first installed the AWS S3 extension into pgAdmin, which allowed us to invoke import, authentication, and URI handling methods. The following is the query we used for each table - it selects only the columns populated with custom data (because PostGres automatically populates the primary key IDs), links to the location of the bucket we created (and the server it's on), and then provides optional authentication details. We specify the default format for our CSV.



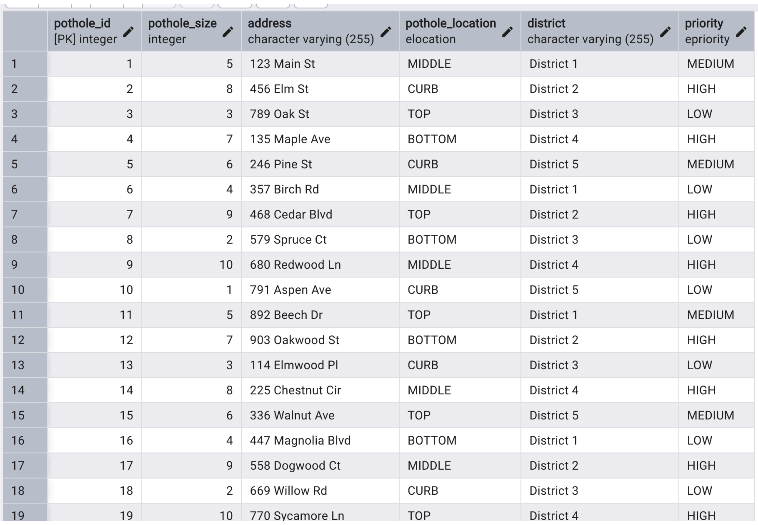
```
SELECT aws_s3.table_import_from_s3('Crew',
'crew_name, crew_size',
'(format csv, header true)',
aws_commons.create_s3_uri('p3-csv', 'csv/crew.csv', 'us-east-2'), aws_commons.create_aws_credentials('AKIA
```

I generated authentication credentials and granted S3 and RDS administrator access to my user account. This is also something we wouldn't want to do in production - instead, we'd want to use IAM. I created a role and policy to grant S3 creation and read permissions. I then went to set this in the RDS instance - however, the interface did not allow me to use my custom role.

Once I settled on using credentials, the above query ran well on each of the tables:



	workorder_id [PK] integer	pothole_id integer	crew_id integer	pothole_status epotstat	cost double precision
1	1	1	1	IN_PROGRESS	500
2	2	2	2	REPAIRED	1200
3	3	3	3	NOT_REPAIRED	300
4	4	4	4	IN_PROGRESS	700
5	5	5	5	REPAIRED	1100
6	6	6	6	NOT_REPAIRED	250
7	7	7	7	IN_PROGRESS	650
8	8	8	8	REPAIRED	1400
9	9	9	9	NOT_REPAIRED	200
10	10	10	10	IN_PROGRESS	800
11	11	11	11	REPAIRED	900
12	12	12	12	NOT_REPAIRED	450
13	13	13	13	IN_PROGRESS	750
14	14	14	14	REPAIRED	1300
15	15	15	15	NOT_REPAIRED	350
16	16	16	16	IN_PROGRESS	[illegible]



	pothole_id [PK] integer	pothole_size integer	address character varying (255)	pothole_location elocation	district character varying (255)	priority epriority
1	1	5	123 Main St	MIDDLE	District 1	MEDIUM
2	2	8	456 Elm St	CURB	District 2	HIGH
3	3	3	789 Oak St	TOP	District 3	LOW
4	4	7	135 Maple Ave	BOTTOM	District 4	HIGH
5	5	6	246 Pine St	CURB	District 5	MEDIUM
6	6	4	357 Birch Rd	MIDDLE	District 1	LOW
7	7	9	468 Cedar Blvd	TOP	District 2	HIGH
8	8	2	579 Spruce Ct	BOTTOM	District 3	LOW
9	9	10	680 Redwood Ln	MIDDLE	District 4	HIGH
10	10	1	791 Aspen Ave	CURB	District 5	LOW
11	11	5	892 Beech Dr	TOP	District 1	MEDIUM
12	12	7	903 Oakwood St	BOTTOM	District 2	HIGH
13	13	3	114 Elmwood Pl	CURB	District 3	LOW
14	14	8	225 Chestnut Cir	MIDDLE	District 4	HIGH
15	15	6	336 Walnut Ave	TOP	District 5	MEDIUM
16	16	4	447 Magnolia Blvd	BOTTOM	District 1	LOW
17	17	9	558 Dogwood Ct	MIDDLE	District 2	HIGH
18	18	2	669 Willow Rd	CURB	District 3	LOW
19	19	10	770 Sycamore Ln	TOP	District 4	HIGH