

Cloud Computing Project Design and Implementation (Group 7)

Ting Han Ko, Xiangxu Li, Yilin Yang

Data Source and Procurement

Data source link:

<https://www.opendatanetwork.com/dataset/mydata.iowa.gov/m3tr-qhgy>

This dataset contains the spirits purchase information of Iowa Class E liquor licensees by product and date of purchase from 2012 to 2022. Class E liquor is unopened liquor containers typically sold in grocery, liquor, and convenience stores. Data was extracted from the open data website, and it was uploaded to our S3 bucket through Amazon S3 console. A csv file originally 4.77 GB, resized to 1.2 GB for performance and efficiency purposes. It contains sales records in grocery stores, liquor stores, and convenience stores.

Amazon VPC

We wanted to set up 2 availability zones each with two subnets for private and public use. Therefore we also need to set up a NAT gateway to connect those subnets together. Below is how we configure our VPC, subnets, route tables, NAT Gateway:

VPC Details Screenshot

The screenshot shows the AWS VPC Details page. On the left, there's a sidebar with navigation links like VPC dashboard, EC2 Global View, Filter by VPC, and a list of services including New VPC Experience, Subnets, Route tables, Internet gateways, Egress-only internet gateways, Carrier gateways, DHCP Option Sets, Elastic IPs, Managed prefix lists, Endpoints, Endpoint services, and NAT gateways. The main content area has a header "Your VPCs (1/2) Info" with a search bar and filter options. A table lists two VPCs:

Name	VPC ID	State	IPv4 CIDR	IPv6 CIDR
IowaProject-vpc	vpc-0e911489219afedc2	Available	10.0.0.0/16	-
-	vpc-0b1eaf74f815af97a	Available	172.31.0.0/16	-

Below the table, a "Details" section provides more information for the selected VPC (IowaProject-vpc):

VPC ID	State	DNS hostnames	DNS resolution
vpc-0e911489219afedc2	Available	Enabled	Enabled
Tenancy	DHCP option set	Main route table	Main network ACL
Default	dopt-028ec9ae455e0db1f	rtb-0f9e07507b948b362	acl-0288bb6351154a2a4
Default VPC	IPv4 CIDR	IPv6 pool	IPv6 CIDR (Network border group)
No	10.0.0.0/16	-	-
Route 53 Resolver DNS Firewall rule groups	Owner ID		
Failed to load rule groups	783113345059		

Two route tables

- IowaProject-rtb-public
- IowaProject-rtb-private1-us-east-1a

Route tables (4) Info							
	Name	Route table ID	Explicit subnet associat...	Edge associations	Main	VPC	Actions Create route table
<input type="checkbox"/>	-	rtb-0c1b917d5d15bea0a	-	-	Yes	vpc-0b1eaf74f8	
<input type="checkbox"/>	-	rtb-0f9e07507b948b362	-	-	Yes	vpc-0e9114892	
<input type="checkbox"/>	IowaProject-rtb-public	rtb-0316c4b16531c27ac	2 subnets	-	No	vpc-0e9114892	
<input type="checkbox"/>	IowaProject-rtb-private1-us-east-1a	rtb-047475261eb33ff79	2 subnets	-	No	vpc-0e9114892	

Public route table

rtb-0316c4b16531c27ac / IowaProject-rtb-public				Actions
Details Info				Run Reachability Analyzer X
<p>ⓘ You can now check network connectivity with Reachability Analyzer</p>				
Route table ID	Main	Explicit subnet associations	Edge associations	
rtb-0316c4b16531c27ac	<input checked="" type="checkbox"/> No	2 subnets	-	
VPC	Owner ID			
vpc-0e911489219afedc2 IowaProject-vpc	783113345059			
Routes Subnet associations Edge associations Route propagation Tags				
Routes (2)				
Edit routes				
Destination	Target	Status	Propagated	
0.0.0.0/0	igw-0759b7203d32e815f	Active	No	
10.0.0.0/16	local	Active	No	

Private route table

The screenshot shows the AWS Route Tables page for a specific VPC. At the top, there is a message indicating network connectivity can be checked with the Reachability Analyzer, with a button to "Run Reachability Analyzer". Below this, the "Details" section provides basic information about the route table, including its ID, association status, owner, and explicit subnet associations. The "Routes" tab is selected, displaying two routes: one for 0.0.0.0/0 targeting a NAT gateway and another for 10.0.0.0/16 targeting the local subnet.

Route table ID	Main	Explicit subnet associations	Edge associations
rtb-047475261eb33ff79	No	2 subnets	-

Destination	Target	Status	Propagated
0.0.0.0/0	nat-034859058687a84ed	Active	No
10.0.0.0/16	local	Active	No

NAT Gateways

The screenshot shows the AWS NAT Gateways page. A single public NAT gateway is listed, named "iowaProject-nat-pu...". The "Details" section provides specific configuration details for this gateway, including its ID, ARN, connectivity type (Public), state (Available), and associated elastic IP address.

Name	NAT gateway ID	Connectivity...	State	Elastic IP address
iowaProject-nat-pu...	nat-034859058687a84ed	Public	Available	34.202.181.63

Four subnets

- IowaProject-subnet-private1-us-east-1a
- IowaProject-subnet-public1-us-east-1a
- IowaProject-subnet-private2
- IowaProject-subnet-public2

Subnets (10) Info		Actions ▾	Create subnet
Filter subnets			
<input type="checkbox"/>	Name	Subnet ID	State
<input type="checkbox"/>	IowaProject-subnet-private1-us-east-1a	subnet-07f613210f374031f	Available
<input type="checkbox"/>	IowaProject-subnet-public1-us-east-1a	subnet-0ed400fa3c8e2a4d0	Available
<input type="checkbox"/>	IowaProject-subnet-private2	subnet-01fb6e58f824a083c	Available
<input type="checkbox"/>	IowaProject-subnet-public2	subnet-0a57337500e8b447e	Available
<input type="checkbox"/>	-	subnet-0ca1e16ebffce13b4	Available
<input type="checkbox"/>	-	subnet-079005b5febcd101	Available
<input type="checkbox"/>	-	subnet-0270c785aa77203f4	Available
<input type="checkbox"/>	-	subnet-01f255e27f027e9be	Available
<input type="checkbox"/>	-	subnet-0e41a98c38ceea648	Available
<input type="checkbox"/>	-	subnet-01b8ba7514fad6f73	Available

Private 1 us-east-1a subnet

subnet-07f613210f374031f / IowaProject-subnet-private1-us-east-1a Actions ▾			
Details			
Subnet ID subnet-07f613210f374031f	Subnet ARN arn:aws:ec2:us-east-1:783113345059:subnet/subnet-07f613210f374031f	State Available	IPv4 CIDR 10.0.1.0/24
Available IPv4 addresses 251	IPv6 CIDR -	Availability Zone us-east-1a	Availability Zone ID use1-az6
Network border group us-east-1	VPC vpc-0e911489219afedc2 IowaProject-vpc	Route table rtb-047475261eb33ff79 IowaProject-rtb-private1-us-east-1a	Network ACL acl-0288bb6351154a2a4
Default subnet No	Auto-assign public IPv4 address No	Auto-assign IPv6 address No	Auto-assign customer-owned IPv4 address No
Customer-owned IPv4 pool -	IPv4 CIDR reservations -	IPv6 CIDR reservations -	IPv6 CIDR reservations -
IPv6-only No	Outpost ID -	Resource name DNS A record Disabled	Resource name DNS AAAA record Disabled
DNS64 Disabled	Hostname type IP name Owner 783113345059		

Public 2 us-east-1b subnet (associated with our EC2)

subnet-0a57337500e8b447e / IowaProject-subnet-public2				Actions ▾
Details				
Subnet ID subnet-0a57337500e8b447e	Subnet ARN arn:aws:ec2:us-east-1:783113345059:subnet/subnet-0a57337500e8b447e	State Available	IPv4 CIDR 10.0.2.0/24	
Available IPv4 addresses 250	IPv6 CIDR -	Availability Zone us-east-1b	Availability Zone ID use1-az1	
Network border group us-east-1	VPC vpc-0e911489219afedc2 IowaProject-vpc	Route table rtb-0316c4b16531c27ac IowaProject-rtb-public	Network ACL acl-0288bb6351154a2a4	
Default subnet No	Auto-assign public IPv4 address No	Auto-assign IPv6 address No	Auto-assign customer-owned IPv4 address No	
Customer-owned IPv4 pool -	IPv4 CIDR reservations -	IPv6 CIDR reservations -	IPv6 CIDR reservations -	
IPv6-only No	Outpost ID -	Resource name DNS A record Disabled	Resource name DNS AAAA record Disabled	
DNS64 Disabled	Hostname type IP name	Owner 783113345059		

Public 1 us-east-1a subnet

subnet-0ed400fa3c8e2a4d0 / IowaProject-subnet-public1-us-east-1a				Actions ▾
Details				
Subnet ID subnet-0ed400fa3c8e2a4d0	Subnet ARN arn:aws:ec2:us-east-1:783113345059:subnet/subnet-0ed400fa3c8e2a4d0	State Available	IPv4 CIDR 10.0.0.0/24	
Available IPv4 addresses 250	IPv6 CIDR -	Availability Zone us-east-1a	Availability Zone ID use1-az6	
Network border group us-east-1	VPC vpc-0e911489219afedc2 IowaProject-vpc	Route table rtb-0316c4b16531c27ac IowaProject-rtb-public	Network ACL acl-0288bb6351154a2a4	
Default subnet No	Auto-assign public IPv4 address No	Auto-assign IPv6 address No	Auto-assign customer-owned IPv4 address No	
Customer-owned IPv4 pool -	IPv4 CIDR reservations -	IPv6 CIDR reservations -	IPv6 CIDR reservations -	
IPv6-only No	Outpost ID -	Resource name DNS A record Disabled	Resource name DNS AAAA record Disabled	
DNS64 Disabled	Hostname type IP name	Owner 783113345059		

Private 2 us-east-1b subnet

subnet-01fb6e58f824a083c / IowaProject-subnet-private2				Actions ▾
Details				
Subnet ID 🔗 subnet-01fb6e58f824a083c	Subnet ARN 🔗 arn:aws:ec2:us-east-1:783113345059:subnet/subnet-01fb6e58f824a083c	State 🕒 Available	IPv4 CIDR 🔗 10.0.3.0/24	
Available IPv4 addresses 🔗 250	IPv6 CIDR –	Availability Zone 🔗 us-east-1b	Availability Zone ID 🔗 use1-az1	
Network border group 🔗 us-east-1	VPC vpc-0e911489219afedc2 IowaProject-vpc	Route table rtb-047475261eb33ff79 IowaProject-rtb-private1-us-east-1a	Network ACL acl-0288bb6351154a2a4	
Default subnet No	Auto-assign public IPv4 address No	Auto-assign IPv6 address No	Auto-assign customer-owned IPv4 address No	
Customer-owned IPv4 pool –	Outpost ID –	IPv4 CIDR reservations –	IPv6 CIDR reservations –	
IPv6-only No	Hostname type IP name	Resource name DNS A record Disabled	Resource name DNS AAAA record Disabled	
DNS64 Disabled	Owner 🔗 783113345059			

Security Groups

We set up security groups to control access to our database, web server etc. They are mostly set to public for now for project sharing purposes. However, in the future we should especially modify Inbound Rules so we can have a better security outcome. Here's how we set up our security groups for now:

1. Web security group

Details

Security group name Web Security Group	Security group ID sg-0056843bd32c71bbd	Description Enable HTTP access	VPC ID vpc-0e911489219afedc2
Owner 783113345059	Inbound rules count 3 Permission entries	Outbound rules count 1 Permission entry	

Inbound rules **Outbound rules** **Tags**

You can now check network connectivity with Reachability Analyzer

Run Reachability Analyzer X

Inbound rules (3)

Type	Protocol	Port range	Source	Description
HTTP	TCP	80	0.0.0.0/0	Permit web requests
All TCP	TCP	0 - 65535	sg-05764f46632cabf7...	Permit Glue requests
SSH	TCP	22	0.0.0.0/0	Permit SSH request

2. DB Security group (for our RDS - PostgresQL Database)

Details

Security group name DB Security Group	Security group ID sg-05764f46632cabf7e	Description Permit access from Web Security Group	VPC ID vpc-0e911489219afedc2
Owner 783113345059	Inbound rules count 2 Permission entries	Outbound rules count 1 Permission entry	

Inbound rules **Outbound rules** **Tags**

You can now check network connectivity with Reachability Analyzer

Run Reachability Analyzer X

Inbound rules (2)

Type	Protocol	Port range	Source	Description
PostgreSQL	TCP	5432	sg-0056843bd32c71b...	-
All TCP	TCP	0 - 65535	sg-05764f46632cabf7...	permit glue access

Amazon EC2

We have to set up an EC2 instance in our VPC for web server usage and temporarily for our data pipeline transitioning data from S3 to our database. In an ideal scenario we will only have to use an EC2 instance solely for the web server most of the time..

Details and Summary Screenshots

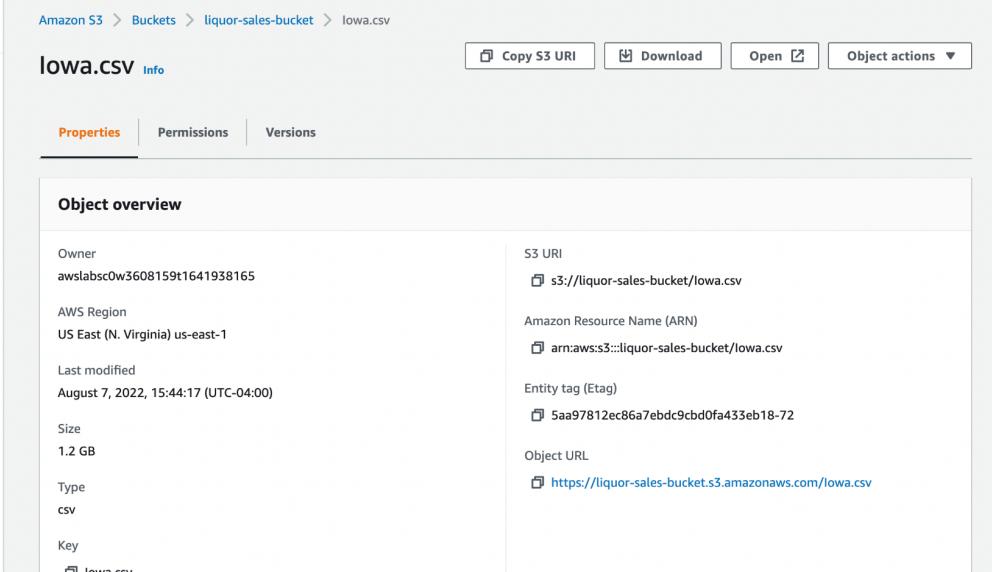
The screenshot shows the AWS EC2 Instances page. On the left, there's a sidebar with navigation links like EC2 Dashboard, EC2 Global View, Events, Tags, Limits, Instances (with 'Instances New'), Instance Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Scheduled Instances, Capacity Reservations, Images (with 'AMIs New'), and AMI Catalog. The main content area has a header 'Instances (1/1) Info' with filters for 'Instance state = running'. It lists one instance: 'Web Server 1' (i-0f92a1a72b3c9b69e). The instance details show it's running, has an instance type t2.micro, and two status checks passed. Below this is a detailed view for 'Instance: i-0f92a1a72b3c9b69e (Web Server 1)'. It includes tabs for Details, Security, Networking, Storage, Status checks, Monitoring, and Tags. Under 'Details', there's a 'Instance summary' section with fields for Instance ID (i-0f92a1a72b3c9b69e), Public IPv4 address (44.199.242.102), Private IPv4 addresses (10.0.2.167), Public IPv4 DNS (ec2-44-199-242-102.compute-1.amazonaws.com), and other networking details.

This is a detailed summary for the same instance. It lists various attributes: Instance ID (i-0f92a1a72b3c9b69e), Public IPv4 address (44.199.242.102), Private IPv4 addresses (10.0.2.167), Public IPv4 DNS (ec2-44-199-242-102.compute-1.amazonaws.com), Instance state (Running), Hostname type (IP name: ip-10-0-2-167.ec2.internal), Private IP DNS name (IPv4 only) (ip-10-0-2-167.ec2.internal), Instance type (t2.micro), Elastic IP addresses (none), Answer private resource DNS name (IPv4 (A)), Auto-assigned IP address (44.199.242.102 [Public IP]), IAM Role (LabRole), VPC ID (vpc-0e911489219afedc2 (IowaProject-vpc)), Subnet ID (subnet-0a57337500e8b447e (IowaProject-subnet-public2)), AWS Compute Optimizer finding (Opt-in to AWS Compute Optimizer for recommendations), and Auto Scaling Group name (none).

Amazon S3

S3 is scalable and reliable for our storage infrastructure. In the case of an extended project, we will be able to scale our storage seamlessly and in a long term project we can even categorize data into different storage trench/levels, to minimize cost. Here's how we set up our S3 bucket for Iowa dataset storage:

S3 Bucket created and the Iowa.csv file uploaded to the S3 bucket

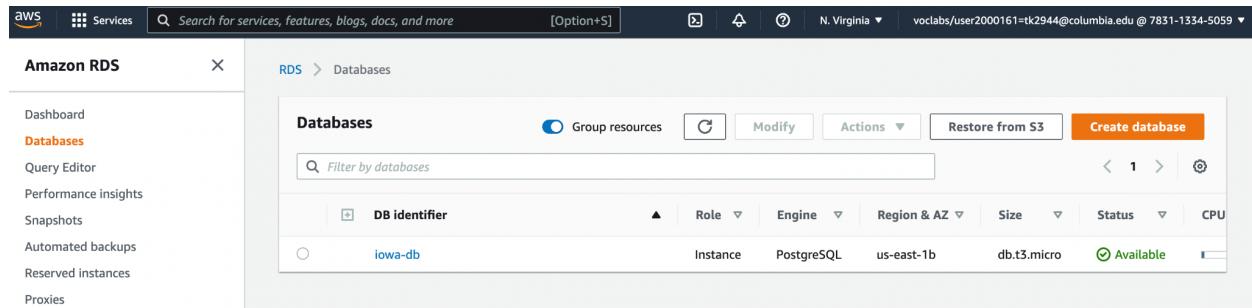


The screenshot shows the Amazon S3 console interface. On the left, there is a sidebar with various navigation options like Buckets, Storage Lens, and AWS Marketplace. The main area displays the properties of an object named 'Iowa.csv' located in the 'liquor-sales-bucket'. The 'Properties' tab is selected, showing details such as Owner (awslabsc0w3608159t1641938165), AWS Region (US East (N. Virginia) us-east-1), Last modified (August 7, 2022, 15:44:17 (UTC-04:00)), Size (1.2 GB), Type (csv), and Key (Iowa.csv). To the right, there are links for S3 URI (s3://liquor-sales-bucket/Iowa.csv), ARN (arn:aws:s3:::liquor-sales-bucket/Iowa.csv), Entity tag (Etag) (Saa97812ec86a7ebdc9cbd0fa433eb18-72), and Object URL (<https://liquor-sales-bucket.s3.amazonaws.com/Iowa.csv>). At the top right, there are buttons for Copy S3 URI, Download, Open, and Object actions.

Amazon RDS

We wanted to set up a database to allow our query to directly access the data and show the query and analysis result, and we wanted to utilize the convenience and interface of Flask building on this database. Here's how we set up the database:

DB Instance



The screenshot shows the Amazon RDS console with the 'Databases' page selected. The left sidebar has links for Dashboard, Databases (which is highlighted in orange), Query Editor, Performance insights, Snapshots, Automated backups, Reserved instances, and Proxies. The main area shows a table titled 'Databases' with one row. The row contains the DB identifier 'iowa-db', the Instance type 'PostgreSQL', the Region & AZ 'us-east-1b', the Size 'db.t3.micro', and the Status 'Available'. There is also a green checkmark icon next to the status.

DB identifier	Instance	Engine	Region & AZ	Size	Status
iowa-db	PostgreSQL	us-east-1b	db.t3.micro	Available	

DB Connectivity and Security (Username: postgres; Password: postgres)

Connectivity & security	Monitoring	Logs & events	Configuration	Maintenance & backups	Tags
Connectivity & security					
Endpoint & port		Networking		Security	
Endpoint iowa-db.casx0mxrevwl.us-east-1.rds.amazonaws.com		Availability Zone us-east-1b		VPC security groups DB Security Group (sg-05764f46632cabf7e) Active	
Port 5432		VPC IowaProject-vpc (vpc-0e911489219afedc2)		Publicly accessible Yes	
		Subnet group db-subnet-group		Certificate authority rds-ca-2019	
		Subnets subnet-01fb6e58f824a083c subnet-07f613210f374031f		Certificate authority date August 22, 2024, 01:08 (UTC±1:08)	
		Network type IPv4			

DB Configuration (database name: Iowa)

Connectivity & security	Monitoring	Logs & events	Configuration	Maintenance & backups	Tags
Instance					
Configuration		Instance class		Storage	
DB instance ID iowa-db		Instance class db.t3.micro		Encryption Not enabled	Performance Insights Performance Insights enabled Turned on
Engine version 14.2		vCPU 2		Storage type General Purpose SSD (gp2)	AWS KMS key aws/rds
DB name Iowa		RAM 1 GB		Storage 20 GiB	Retention period 7 days
License model Postgresql License	Availability			Provisioned IOPS -	Database activity stream
Option groups default:postgres-14 In sync	Master username postgres			Storage autoscaling Enabled	Status Stopped
Amazon Resource Name (ARN) arn:aws:rds:us-east-1:783113345059:db:iowa-db	IAM DB authentication Not enabled			Maximum storage threshold 1000 GiB	Audit policy status -
Resource ID db-	Multi-AZ No				
	Secondary zone Casx0mxrevwl.us-east-1.rds.amazonaws.com				

AWS IAM

We attached policies to LabRole in IAM to allow EC2 and RDS to access S3 so we can continue our ETL process. However, in the future we should set up more roles to ensure a better divided task access from different groups, enhancing our security. For now, this is how we set up our IAM role:

The screenshot shows the AWS IAM Roles experience interface. On the left, a sidebar navigation includes: Dashboard, Access management (User groups, Users, Policies, Identity providers, Account settings), Access reports (Access analyzer, Archive rules, Analyzers, Settings, Credential report, Organization activity, Service control policies (SCPs)), and Related consoles (IAM Identity Center). The main content area displays the 'LabRole' summary, which includes creation date (June 18, 2022, 20:42 UTC-04:00), ARN (arn:aws:iam::783113345059:role/LabRole), last activity (18 minutes ago), maximum session duration (1 hour), and instance profile ARN (arn:aws:iam::783113345059:instance-profile/LabInstanceProfile). Below the summary are tabs for Permissions, Trust relationships, Tags (1), Access Advisor, and Revoke sessions. The 'Permissions' tab is selected, showing 8 managed policies attached to the role. The policies listed are:

Policy name	Type	Description
c55725a929065122947581w783113345059-VocLabPolicy1-1DXIU5ARTEA1E	Customer managed	
c55725a929065122947581w783113345059-VocLabPolicy2-3ZGZLHYJMTHU	Customer managed	
c55725a929065122947581w783113345059-VocLabPolicy3-15K7LXJ5C9XXB	Customer managed	
AmazonRDSFullAccess	AWS managed	Provides full access
AmazonS3FullAccess	AWS managed	Provides full access
AmazonRDSDataServiceAccess	AWS managed	Allow RDS to access
AmazonSSMManagedInstanceCore	AWS managed	The policy for Amazon
AmazonRDSDynamoDBFullAccess	AWS managed	Allows full access to

ETL Pipeline

We directly transfer the data through the command line interface from S3 Bucket to our EC2 Instance, and then from EC2 instance to our RDS DB Iowa database.

Screenshot of Importing the “Iowa.csv” file from S3 to EC2

```
[ec2-user@ip-10-0-2-167 ~]$ aws s3 cp "s3://liquor-sales-bucket/Iowa.csv" Iowa.csv
download: s3://liquor-sales-bucket/Iowa.csv to ./Iowa.csv
[ec2-user@ip-10-0-2-167 ~]$ ls
Iowa.csv  Iowa_Liquor_Sales.csv  out.csv  testingImport.py  testing.py  tiny_file.csv
```

Create an empty table in RDS DB- Iowa through EC2

```
[ec2-user@ip-10-0-2-167 ~]$ psql Iowa    -U postgres    -p 5432    -h iowa-db.casx0mxrevwl.us-east-1.rds.amazonaws.com    -c "DROP TABLE IF EXISTS sales CASCADE;"  
Password for user postgres:  
DROP TABLE  
[ec2-user@ip-10-0-2-167 ~]$ psql Iowa    -U postgres    -p 5432    -h iowa-db.casx0mxrevwl.us-east-1.rds.amazonaws.com    -c "CREATE TABLE sales(  
>     InvoiceItemNumber VARCHAR(50),\n>     Date DATE,\n>     StoreNumber FLOAT,\n>     StoreName VARCHAR(50),\n>     Address VARCHAR(100),\n>     City VARCHAR(50),\n>     ZipCode VARCHAR(10),\n>     StoreLocation VARCHAR(100),\n>     CountyNumber FLOAT,\n>     County VARCHAR(50),\n>     Category FLOAT,\n>     CategoryName VARCHAR(50),\n>     VendorNumber FLOAT,\n>     VendorName VARCHAR(100),\n>     ItemNumber VARCHAR(50),\n>     ItemDescription VARCHAR(100),\n>     Pack FLOAT,\n>     BottleVolumeML FLOAT,\n>     StateBottleCost numeric(20,2),\n>     StateBottleRetail numeric(20,2),\n>     BottlesSoldInMILAT,\n>     SalesInDollars numeric(20,2),\n>     VolumeSoldInLiters numeric(20,2),\n>     VolumeSoldInGallons numeric(20,2),\n>     PRIMARY KEY(InvoiceItemNumber)\n> )  
> "  
Password for user postgres:  
CREATE TABLE
```

Code reference:

```
psql Iowa    -U postgres    -p 5432    -h  
iowa-db.casx0mxrevwl.us-east-1.rds.amazonaws.com    -c "DROP TABLE IF EXISTS  
sales CASCADE;"
```

```
psql Iowa    -U postgres    -p 5432    -h  
iowa-db.casx0mxrevwl.us-east-1.rds.amazonaws.com    -c "CREATE TABLE sales(  
    InvoiceItemNumber VARCHAR(50),\n    Date DATE,\n    StoreNumber FLOAT,\n    StoreName VARCHAR(50),\n    Address VARCHAR(100),\n    City VARCHAR(50),\n    ZipCode VARCHAR(10),\n    StoreLocation VARCHAR(100),\n    CountyNumber FLOAT,\n    County VARCHAR(50),\n    Category FLOAT,\n    CategoryName VARCHAR(50),\n    VendorNumber FLOAT,\n    VendorName VARCHAR(100),\n    ItemNumber VARCHAR(50),\n    ItemDescription VARCHAR(100),\n    Pack FLOAT,\n    BottleVolumeML FLOAT,\n    StateBottleCost numeric(20,2),\n    StateBottleRetail numeric(20,2),\n    BottlesSoldInMILAT,\n    SalesInDollars numeric(20,2),\n    VolumeSoldInLiters numeric(20,2),\n    VolumeSoldInGallons numeric(20,2),\n    PRIMARY KEY(InvoiceItemNumber)
```

```

County VARCHAR(50),\
Category FLOAT,\n
CategoryName VARCHAR(50),\n
VendorNumber FLOAT,\n
VendorName VARCHAR(100),\n
ItemNumber VARCHAR(50),\n
ItemDescription VARCHAR(100),\n
Pack FLOAT,\n
BottleVolumeML FLOAT,\n
StateBottleCost numeric(20,2),\n
StateBottleRetail numeric(20,2),\n
BottlesSold FLOAT,\n
SaleInDollars numeric(20,2),\n
VolumeSoldInLiters numeric(20,2),\n
VolumeSoldInGallons numeric(20,2),\n
PRIMARY KEY(InvoiceItemNumber)\n
)

```

Loaded the csv file from S3 to RDS DB- Iowa and read table (first 5 rows)

```

[ec2-user@ip-10-0-2-167 ~]$ psql Iowa -U postgres -p 5432 -h iowa-db.casx0mxrevwl.us-east-1.rds.amazonaws.com -c "\copy sales from 'Iowa.csv' WITH DELIMITER ',' CSV HEADER;\n"
Password for user postgres:
COPY 4999999
[ec2-user@ip-10-0-2-167 ~]$ psql Iowa -U postgres -p 5432 -h iowa-db.casx0mxrevwl.us-east-1.rds.amazonaws.com -c "SELECT * FROM sales LIMIT 5"
Password for user postgres:
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| invoiceitemnumber | date | storenumber | storename | address | city | zipcode | storelocation | countynumber | county |
| category | categoryname | vendornumber | vendorname | itemnumber | itemdescription | pack | bottlevolume | statebottlecost | statebottler |
| etall | bottlesold | saleindollars | volumesoldinliters | volumesoldingallons |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| S15062208002 | 2013-10-10 | 2633 | Hy-Vee #3 / BDI / Des Moines | 3221 SE 14TH ST | DES MOINES | 50320 | POINT (-93.596754 41.554101) | 77 | Polk |
| 1082900 | MISC. IMPORTED CORDIALS & LIQUEURS | 305 | MHW Ltd | 904969 | Sabe Premium Sake Double Barrel | 6 | 750 | 14.99 |
22.49 | 6 | 134.94 | 4.50 | 1.19 |
| S19323500038 | 2014-06-03 | 2687 | Hy-Vee Wine and Spirits / Shenandoah | 520 SO FREMONT | SHENANDOAH | 51601 | POINT (-95.385111 40.761736) | 73 | Page |
| 1862200 | PUERTO RICO & VIRGIN ISLANDS RUM | 434 | Luxco-St Louis | 45277 | Paramount White Rum | 12 | 1000 | 4.34 |
6.51 | 12 | 78.12 | 12.00 | 3.17 |
| S23334500013 | 2015-01-06 | 4810 | Kum & Go #518 / Ankeny | 3603 NE OTTERVIEW CIRCLE | ANKENY | 50021 | POINT (-93.572458 41.768989) | 77 | Polk |
| 1862200 | PUERTO RICO & VIRGIN ISLANDS RUM | 35 | Bacardi U.S.A., Inc. | 43121 | Bacardi Superior Rum Mini | 12 | 500 | 5.54 |
8.31 | 1 | 8.31 | 0.50 | 0.13 |
| S09742200010 | 2012-12-27 | 4025 | Karam Kaur Khasriya Llc | 702 137H ST | BELLE PLAINE | 52208 | POINT (-92.277759 41.897052) | 6 | Benton |
| 1812100 | CANADIAN WHISKIES | 260 | Diageo Americas | 11298 | Crown Royal Canadian Whisky | 6 | 1750 | 31.00 |
46.49 | 2 | 92.98 | 3.50 | 0.92 |
| S15034600007 | 2013-10-09 | 4583 | Kum & Go #5100 / Manson | 208 MAIN ST | MANSON | 50563 | POINT (-94.534532 42.517855) | 13 | Calhou |
n | 1081200 | CREAM LIQUEURS | 305 | MHW Ltd | 73050 | Rumchata | 6 | 750 | 12.50 |
18.75 | 6 | 112.50 | 4.50 | 1.19 |
(5 rows)

[ec2-user@ip-10-0-2-167 ~]$ 

```

Code reference:

```

psql Iowa -U postgres -p 5432 -h
iowa-db.casx0mxrevwl.us-east-1.rds.amazonaws.com\
-c "\COPY sales FROM 'Iowa.csv' WITH DELIMITER ',' CSV HEADER;"
```

```
psql Iowa -U postgres -p 5432 -h iowa-db.casx0mxrevwl.us-east-1.rds.amazonaws.com  
-c "SELECT * FROM sales LIMIT 5"
```

Data Analysis Example

```
[ec2-user@ip-10-0-2-167 ~]$ psql Iowa -U postgres -p 5432 -h iowa-db.casx0mxrevwl.us-east-1.rds.amazonaws.com -c "select county, sum(SaleinDollars) as total_sales_in_dollars  
> from sales  
> group by County  
> order by total_sales_in_dollars desc  
> limit 10;  
>"  
Password for user postgres:  
          county | total_sales_in_dollars  
          Polk    |      139804986.82  
          Linn    |      55856658.88  
          Scott   |      45885159.35  
          Johnson |      38492619.19  
          Black Hawk |      36977974.68  
          Pottawattamie |      22662816.67  
          Woodbury |      21766637.46  
          Story   |      19591146.13  
          Dubuque |      19028937.38  
          Cerro Gordo |      13099243.90  
(10 rows)
```

Code reference:

```
psql Iowa -U postgres -p 5432 -h iowa-db.casx0mxrevwl.us-east-1.rds.amazonaws.com  
-c "select county, sum(SaleinDollars) as total_sales_in_dollars  
from sales  
group by County  
order by total_sales_in_dollars desc  
limit 10;
```

Flask Server

We wanted to build a flask server to show query and simple analysis results.
Example of using python file to activate flask and query/ analyze data:

```

^C[ec2-user@ip-10-0-2-167 ~]$ sudo nano query_analysis_server.py
[ec2-user@ip-10-0-2-167 ~]$ sudo python3 query_analysis_server.py
[WARNING: This is a development server. Do not use it in a production deployment. Use a productio]
[n WSGI server instead.
 * Serving Flask app 'APAN5450App'
 * Debug mode: off
 * Running on all addresses (0.0.0.0)
   WARNING: This is a development server. Do not use it in a production deployment.
 * Running on http://127.0.0.1:80
 * Running on http://10.0.2.167:80 (Press CTRL+C to quit)
158.222.128.225 - - [10/Aug/2022 01:56:25] "GET / HTTP/1.1" 200 -
^C[ec2-user@ip-10-0-2-167 ~]$ sudo nano query_analysis_server.py
[ec2-user@ip-10-0-2-167 ~]$ sudo python3 query_analysis_server.py
[WARNING: This is a development server. Do not use it in a production deployment. Use a productio]
[n WSGI server instead.
 * Serving Flask app 'APAN5450App'
 * Debug mode: off
 * Running on all addresses (0.0.0.0)
   WARNING: This is a development server. Do not use it in a production deployment.
 * Running on http://127.0.0.1:80
 * Running on http://10.0.2.167:80 (Press CTRL+C to quit)
158.222.128.225 - - [10/Aug/2022 01:57:29] "GET / HTTP/1.1" 200 -

```

Result showing on the web page:

```

[The number of rows: 10,
county': 'Polk', 'total_sales_in_dollars': Decimal('139006906.02'),
county': 'Linn', 'total_sales_in_dollars': Decimal('55856650.00'),
county': 'Scott', 'total_sales_in_dollars': Decimal('45085159.35'),
county': 'Johnson', 'total_sales_in_dollars': Decimal('38492619.19'),
county': 'Black Hawk', 'total_sales_in_dollars': Decimal('36977974.60'),
county': 'Pottawattamie', 'total_sales_in_dollars': Decimal('22662816.67'),
county': 'Woodbury', 'total_sales_in_dollars': Decimal('21766637.46'),
county': 'Story', 'total_sales_in_dollars': Decimal('19591146.13'),
county': 'Dubuque', 'total_sales_in_dollars': Decimal('19020937.30'),
county': 'Cerro Gordo', 'total_sales_in_dollars': Decimal('13099243.90')]

```

Flask Code Reference:

```

import sys
import psycopg
sys.path.append("/home/ec2-user/.local/lib/python3.7/site-packages")
from flask import Flask

app = Flask("APAN5450App")

@app.route("/")
def hello_world():

```

```

output = []
# Connect to an existing database
with psycopg.connect(
    host="iowa-db.casx0mxrevwl.us-east-1.rds.amazonaws.com",
    port="5432",
    dbname="iowa",
    user="postgres",
    password="postgres") as conn:
    # Open a cursor to perform database operations
    with conn.cursor() as cur:
        # Query the database to verify the inserted data
        queryCmd = """SELECT county, sum(SaleinDollars) AS
total_sales_in_dollars FROM sales GROUP BY County ORDER BY
total_sales_in_dollars DESC LIMIT 10;"""
        cur.execute(queryCmd)
        output.append("The number of rows: ")
        output.append(cur.rowcount)

        row = cur.fetchone()
        while row is not None:
            output.append("<br>county:")
            output.append(row[0])
            output.append(", total_sales_in_dollars:")
            output.append(row[1])
            row = cur.fetchone()

        # Close the cursor
        cur.close()
return str(output)

app.run(host='0.0.0.0', port=80)

```

Amazon SageMaker

We wanted to include further analysis possibilities in the future, especially machine learning models for time series analysis to predict sales performance. Therefore, we wanted to set up SageMaker, different from Flask, we will conduct deeper model training on the SageMaker Instance that requires a heavier workload.

We built a notebook instance in AWS Sagemaker, and we could open the Jupyter Lab in this instance. We successfully imported data and used Python to analyze the data in Sagemaker.

The screenshot shows the AWS SageMaker interface. On the left, there's a sidebar with various navigation options like Control panel, Studio, etc. The main content area is titled 'IowaNB' and shows 'Notebook instance settings'. It lists the following details:

Name	Status	Notebook instance type	Platform identifier
IowaNB	InService	mLm4.xlarge	Amazon Linux 2, Jupyter Lab 1 (notebook-al2-v1)
ARN	Creation time	Elastic Inference	Minimum IMDS Version
arn:aws:sagemaker:us-east-1:783113345059:notebook-instance/iowanb	Aug 10, 2022 00:30 UTC	-	1
Lifecycle configuration	Last updated	Volume Size	
-	Aug 10, 2022 00:33 UTC	5GB EBS	

Below this, there's a section for 'Git repositories' which currently has no resources.

Analysis Examples:

Here are some analytical results.

First, we want to figure out which cities sell the most liquor. This could help us decide demands, target cities and amounts we can supply. We find that Des Moines, Cedar Rapids, and Davenport are the top 3 cities in terms of bottles sold. And these cities will be our potential market to enter.

Top 3 cities in terms of amount sold.

```
result1=pd.value_counts(df.city)
print(result1.head(3))
```

DES MOINES	441709
CEDAR RAPIDS	325014
DAVENPORT	222359

We also analyze the sales of each category. According to the result below, we find that Vodka and Whiskies are popular in the Iowa market. We could sell these liquor to Iowa.

Top categories in terms of amount sold

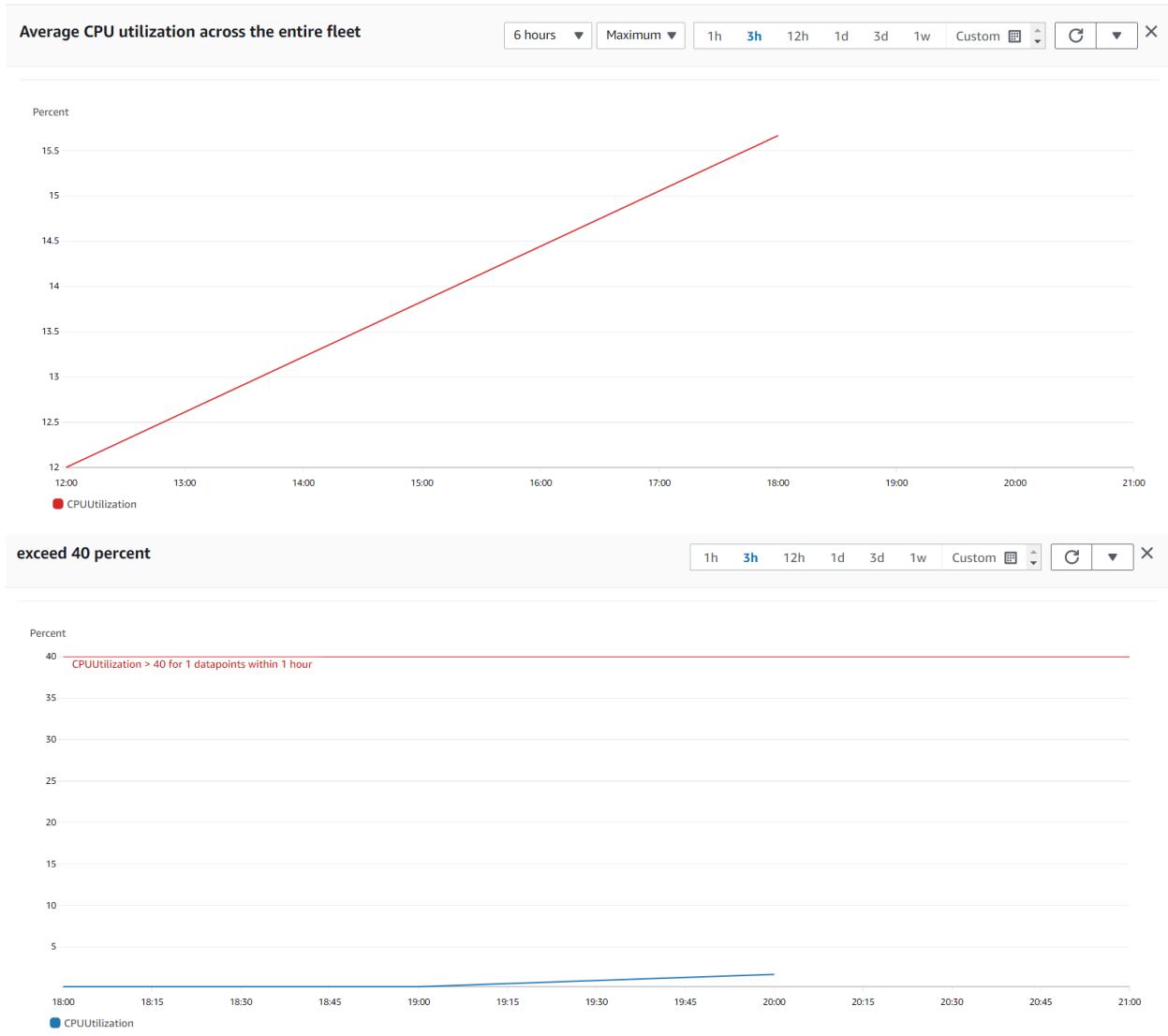
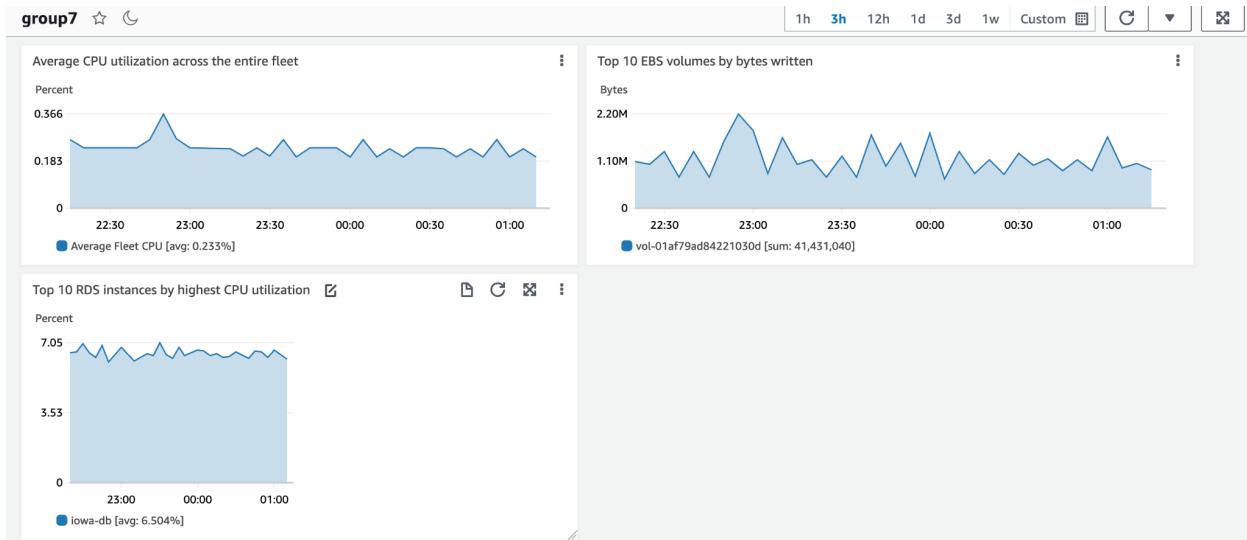
```
pd.value_counts(np.array(df.Category))
```

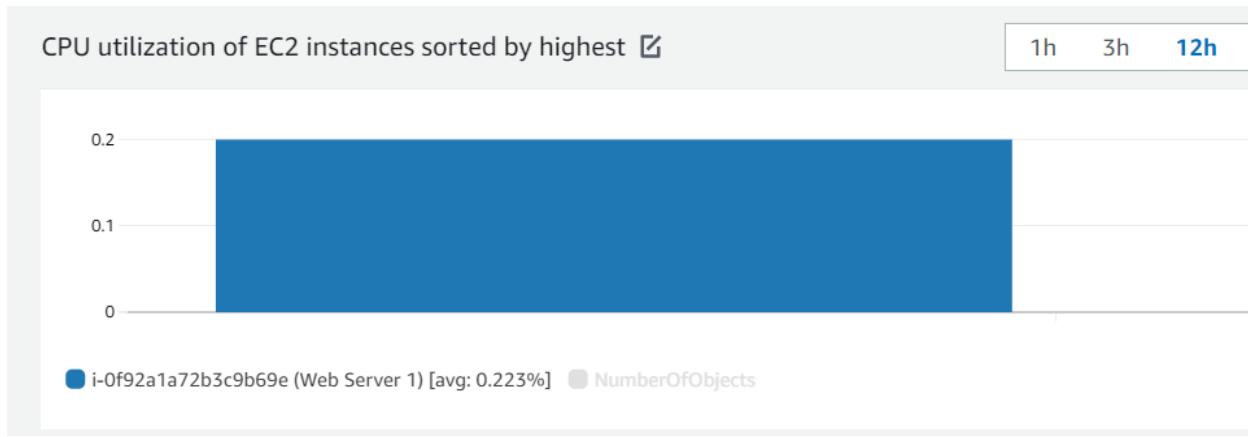
```
1031080.0    625775
1012100.0    469118
1011200.0    273401
1062310.0    267232
1031200.0    251262
...
1092100.0     4
1092000.0     4
101220.0      4
1082300.0     1
1089290.0     1
Length: 104, dtype: int64
```

Category Number	Category Name
1031080	VODKA 80 PROOF
1012100	CANADIAN WHISKIES
1011200	STRAIGHT BOURBON WHISKIES
1062310	SPICED RUM
1031200	VODKA FLAVORED

Cloudwatch dashboard and alert:

We can monitor the CPU utilizations of EC2, EBS volumes, etc. on the Cloudwatch dashboard. An alert is also set for when CPU utilization is higher than expected.(We will receive the alert email if CPU utilization is high)





Cost analysis(1 year)

We choose cloud computing for our project and based on the requirements of analysis, we need these settings and configurations: 1TB Storage, vCPUs: 4, Memory: 16GB, Storage: 1TB. And then we compared Azure with AWS to figure out which one could save money for us.

Azure 1 year Total cost of ownership:

Workload	Environment	Operating system	Operating System License	VMs	Virtualization
Windows/Linux Server	Virtual Machines	Windows	Datacenter	4	Hyper-V
Core(s)	RAM (GB)	Optimize by	Windows Server 2008/2008 R2		
1	16	CPU			

(1 - 32) (1 - 448)

Storage

Enter the details of your on-premises storage infrastructure. After adding storage, select the storage type and enter the remaining details.

Storage 1

Storage type ⓘ	Disk type ⓘ	Capacity ⓘ	Backup ⓘ	Archive ⓘ
Local Disk/SAN	HDD	1 TB (1 - 5000)	1 TB (0 - 5000)	1 TB (0 - 5000)

+ Add storage

Networking

Enter the amount of network bandwidth you currently consume in your on-premises environment.

Outbound bandwidth ⓘ

1
GB

(1 - 2000000)

Cost over 1 year(s)

Azure cost breakdown summary

Category	Cost
Compute	\$1,948.56
Data Center	\$0.00
Networking	\$0.12
Storage	\$831.12
IT Labor	\$1,533.41
Total	\$4,313.00

AWS 1 year Total cost of ownership:

vCPUs 4	GPUs NA
Memory (GiB) 16 GiB	Network performance Up to 5 Gigabit

Storage per instance

Storage for each EC2 instance
Choose EBS volume storage type.

General Purpose SSD (gp2)

Storage amount
1 TB

Estimate summary Info

Upfront cost 689.41 USD	Monthly cost 102.40 USD	Total 12 months cost 1,918.21 USD <small>Includes upfront cost</small>
----------------------------	----------------------------	---

Based on calculations, we find AWS could save more than 50% cost for us if we join EC2 1-Year Instance Savings Plans. That's why we chose AWS for our project.

Success criteria

1. Set up VPC
2. Set up security groups
3. Load data into an S3 bucket
4. Data processed and flask running in EC2
5. Modify a role in IAM to allow EC2 and RDS to access S3
6. RDS configured and loaded with processed data
7. Activate flask server that query data and display analytic results
8. Set up Cloudwatch to monitor CPU utilization and send alarms if CPU utilization is high
9. Set up a notebook instance in Sagemaker to analyze data

Summary

Based on the screenshots and descriptions above, we have completed our clouds computing project and met all of the success criteria. We set up a VPC first, loaded data into an S3 bucket and built a flask server running in EC2 to process and analyze data.

After that, we set up a database on RDS and access the data from both the CLI and flask server to display the data and analysis result. The analysis results could help our client better understand the Iowa liquor market. Besides, we also set up a Python notebook instance in Sagemaker to analyze data. To ensure security and privacy, we modified a role in IAM to allow EC2 and RDS to access S3 . We also set up security groups accordingly.

Also, we set up Cloudwatch to monitor CPU utilizations and send alarms if CPU utilization is higher than we expected. For cost analysis, we compared our cost of cloud computing on AWS and Microsoft Azure. We found that AWS's cost is significantly lower than Azure's cost.