Analyst Workshop

In this workshop, we will be working with some dataset that contains insurance rates data from across the United States, providing insights into the premiums charged by insurers, the underlying factors that affect those rates, and claims history analysis. The data is designed to help researchers understand the inner workings of the insurance industry, and how rates are calculated. It includes information on premiums, underlying factors, current premium prices, indicated premium prices, selected premium prices, fixed expenses, and more

There are 3 tables with the schemas and column definitions below.

- Cgr_definitions_table - Dimension table with the premium rates for a Combined grade rating (CGR)
- Cgr_premiums_table - Historic dataset on customer's premium, expenses and CGR factor
- Territory_definitions_table - Location data such as county, country_code, zipcode related to the customer's territory

Cgr_definitions_table

Column name	Description
cgr	Combined grade rating. (Numeric)
aa	Average annual premium. (Numeric)
bb	Base premium. (Numeric)
cc	Cost of capital. (Numeric)
va	Value of assets. (Numeric)
dd	Direct written premium. (Numeric)
hh	Homeownership. (Categorical)

Cgr_premiums_table

Column name	Description
territory	The territory in which the person lives. (String)
gender	The person's gender. (String)
birthdate	The person's birthdate. (Date)
ypc	The person's years of prior coverage. (Integer)
current_premium	The person's current premium. (Float)
indicated_premium	The person's indicated premium. (Float)
selected_premium	The person's selected premium. (Float)
underlying_premium	The person's underlying premium. (Float)

fixed_expenses	The person's fixed expenses. (Float)
underlying_total_premium	The person's underlying total premium. (Float)
cgr_factor	The person's CGR factor. (Float)

Territory_definitions_table

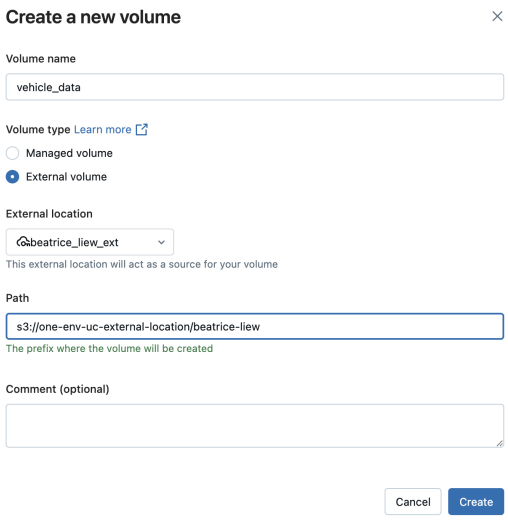
Column name	Description
territory	The territory in which the person lives. (String)
county	The county in which the person lives. (String)
county_code	The county code for the county in which the person lives. (String)
zipcode	The zip code for the county in which the person lives. (String)
town	The town in which the person lives. (String)

Data Upload / Ingestion via UI

1. Create a schema with your name / unique ID
2. Upload each csv via the UI to create a new table within your schema
3. Once the tables are created, navigate to the table and there would be an AI generated description for the table. Accept or edit as necessary
4. This makes it easy for the search function to locate the right data object with the right metadata
5. Search for CGR with the full definition

Data upload via Volumes

1. Besides creating a direct delta table, you can also upload files into a volume as csv. This can be done either via the Databricks UI or a direct upload into ADLS.
2. First create a volume pointing towards an external location that you have set up on Databricks.
3. Create subdirectories within for the different files, as this will future proof it if there are additional new files that should be appended to the table



Data modelling

1. Switch over to the SQL editor
2. Create the primary keys by running the following command. As the columns cannot be nulls for primary keys, we need to set the constraint to non nullable.

```
ALTER TABLE cgr_definitions_table ALTER COLUMN cgr SET NOT NULL;
ALTER TABLE cgr_definitions_table ADD CONSTRAINT cgr_pk PRIMARY KEY(cgr);
ALTER TABLE territory_definitions_table ALTER COLUMN territory SET NOT NULL;
ALTER TABLE territory_definitions_table ADD CONSTRAINT territory_pk PRIMARY
KEY(territory);
```

3. Create the foreign keys by running the following command. This links the foreign keys to the primary keys in the dimension tables

```
ALTER TABLE Cgr_premiums_table ADD CONSTRAINT cgr_fk FOREIGN KEY(cgr) REFERENCES
cgr_definitions_table;
ALTER TABLE Cgr_premiums_table ADD CONSTRAINT territory_fk FOREIGN KEY(territory)
REFERENCES territory_definitions_table;
```

4. Flip back to the catalog view of the tables and there would be a PK / FK icon next to the columns
5. Click on view relationships to see the model / ERD diagram (If available)

Data analysis

1. Return to SQL editor
2. Let's add an age column to the Cgr_premiums_table using the query below (Or try to use Databricks assistant for this if not familiar with the syntax: Return a query to calculate the age in years based on birthdate)

```
SELECT *, datediff(year, birthdate, CURRENT_DATE()) as customer_age
FROM cgr_premiums_table
```

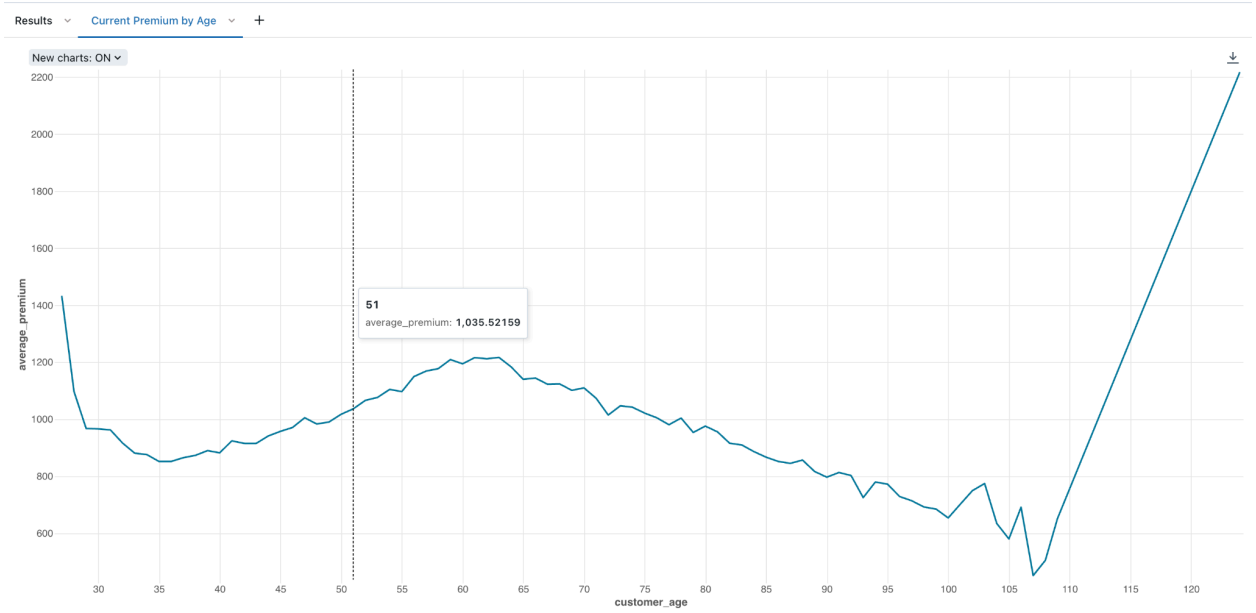
3. Find the trend between average current_premium and the age of the customer (Or ask assistant: what is the relationship between average of current premium and the customer's age)

```
select customer_age, avg(current_premium) as average_premium from (
SELECT *, datediff(year, birthdate, CURRENT_DATE()) as customer_age
FROM cgr_premiums_table)
group by customer_age
order by customer_age asc
```

Results +

#	customer_age	average_premium
1	27	1429.78
2	28	10 1429.78
3	29	966.67
4	30	965.13
5	31	961.63
6	32	915.50
7	33	880.06
8	34	875.09
9	35	851.06
10	36	851.05
11	37	864.15
12	38	872.46
13	39	888.96
14	40	881.24
15	41	923.36
16	42	914.42

4. Create a visualisation to visualise this using a line chart with customer_age on the x axis and average_premium on the y axis.



5. Notice that data seems odd after >100 years. Let's use a where clause to filter it out.

```
select customer_age, avg(current_premium) as average_premium from (
SELECT *, datediff(year, birthdate, CURRENT_DATE()) as customer_age
FROM cgr_premiums_table
)
where customer_age < 100
group by customer_age
order by customer_age asc
```

6. Enrich the historic customer premium information with location data by joining to the territory_definitions_table. Create a view with this.

```
SELECT a.*, datediff(year, a.birthdate, CURRENT_DATE()) as customer_age, b.*
FROM cgr_premiums_table a
LEFT JOIN territory_definitions_table b
on a.territory = b.territory
```

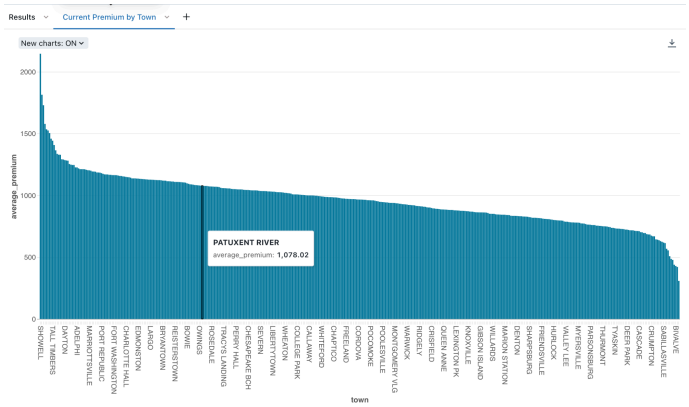
7. Find the average current_premium by town, which town has the highest premium? How many customers are there? (Assistant: What is the average current premium by town, using the table territory_definitions_table. How many customers are in each town, alongside their average premium?)

```
select town, avg(current_premium) as average_premium, count(*) as number_of_customers
from
(
SELECT a.*, datediff(year, a.birthdate, CURRENT_DATE()) as customer_age, b.*
FROM cgr_premiums_table a
LEFT JOIN territory_definitions_table b
on a.territory = b.territory
)
group by town
order by average_premium desc
```

Results | Current Premium by Age | +

#	town	average_premium	number_of_customers
1	SHOWELL	2145.03	1
2	HARMANS	1813.14	1
3	KITZMILLER	1728.24	4
4	BARNESVILLE	1576.99	3
5	DOWELL	1533.72	3
6	STEVENSON	1523.91	7
7	CHELTENHAM	1503.08	64
8	BUCKEYSTOWN	1457.93	2
9	TALL TIMBERS	1441.36	6
10	UPPER FALLS	1406.38	8
11	BEL ALTON	1363.64	16
12	CLARKSVILLE	1334.80	133
13	WELCOME	1328.34	23
14	ISSUE	1327.59	7

8. Create a chart to visualise the relationship between premium and town



9. To reuse the query, save it on SQL editor into your own workspace. You can also share the query with others

Scheduling queries

1. To schedule this query to run at a set time, save the query first.
2. Use the query below to create a view and save the query

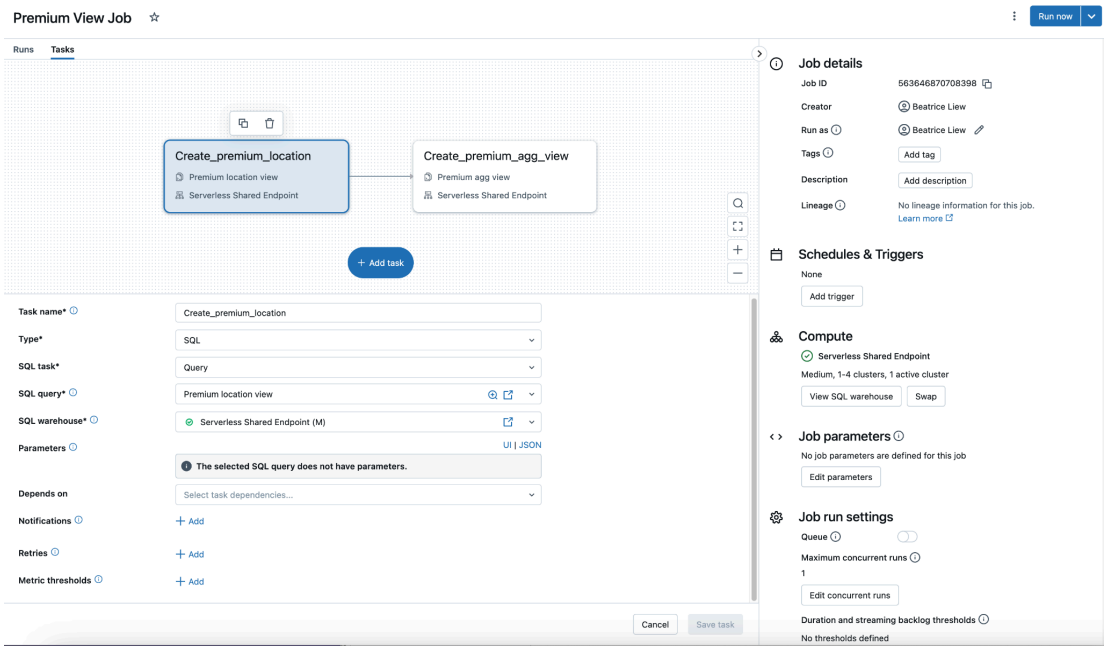
```
CREATE OR REPLACE VIEW cgr_premiums_table_location as
SELECT a.*, datediff(year, a.birthdate, CURRENT_DATE()) as customer_age, b.county,
b.county_code, b.zipcode, b.town, b.area
FROM cgr_premiums_table a
```

```
LEFT JOIN territory_definitions_table b
on a.territory = b.territory
```

3. Create an aggregated view based on this view to get the average premium by town. Save the query

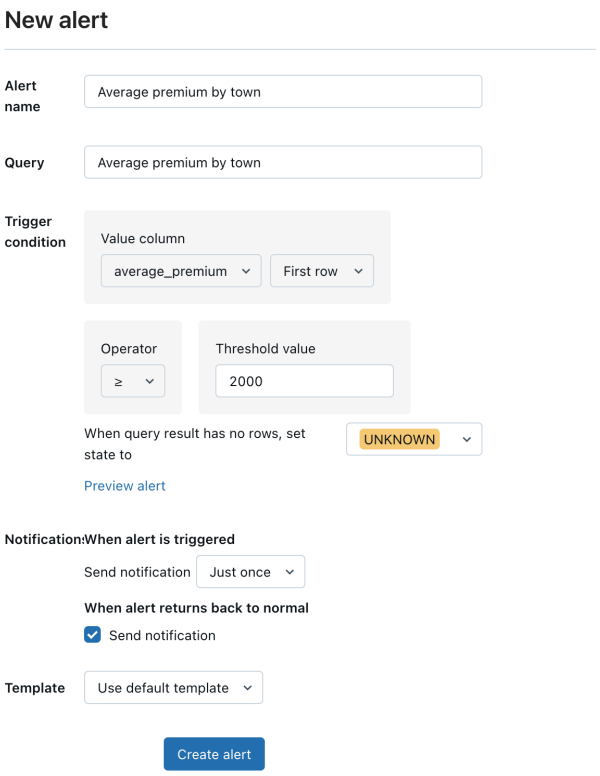
```
CREATE OR REPLACE VIEW cgr_premiums_table_agg as
select town, avg(current_premium) as average_premium, count(*) as number_of_customers
from cgr_premiums_table_location
group by town
order by average_premium desc
```

4. If the SQL queries are not dependent on each other / anything else, click on schedule and set the frequency required.
5. To build dependencies between scheduled queries, use workflows
6. Click create job
7. Click on create task and add the SQL query in for creating cgr_premiums_table_location
8. Add another task and add the SQL query for creating cgr_premiums_table_agg
9. Add in a schedule or trigger for the update
10. There's also an option to add in notification via email to notify in the event of failure / success



Alerts

1. There's an option to create an alert to listen in on the data
2. Let's say we would like to be alerted if there's any town's with an average premium > 2000
3. Set up the alert as shown below and click on create alert



4. Set up a schedule for when you want the check to be done, which would either trigger an alert if the trigger condition is met, or not if not met
5. Add your email as the notification destination
6. Click on run once to test that it is working. You should receive an email when it's triggered

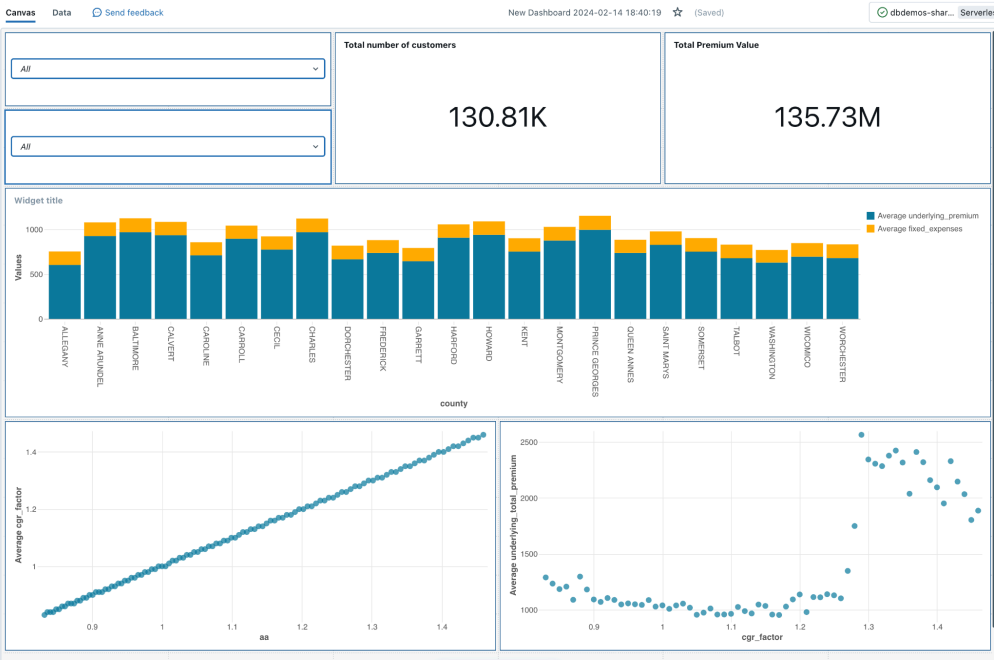
Lakeview dashboards

1. Navigate to Dashboards and click on lakeview dashboard
2. First, add data that we want to visualise in the data tab. Click on select a table and choose the view with data enriched with location data cgr_premiums_table_location
3. There's also an option to define transformations using create from SQL. Enrich the cgr_premiums_table using the table cgr_definitions_table with the query below

```
select *
from beatrice_liew.vehicle_data.cgr_premiums_table a
left join beatrice_liew.vehicle_data.cgr_definitions_table b
on a.cgr = b.cgr
```

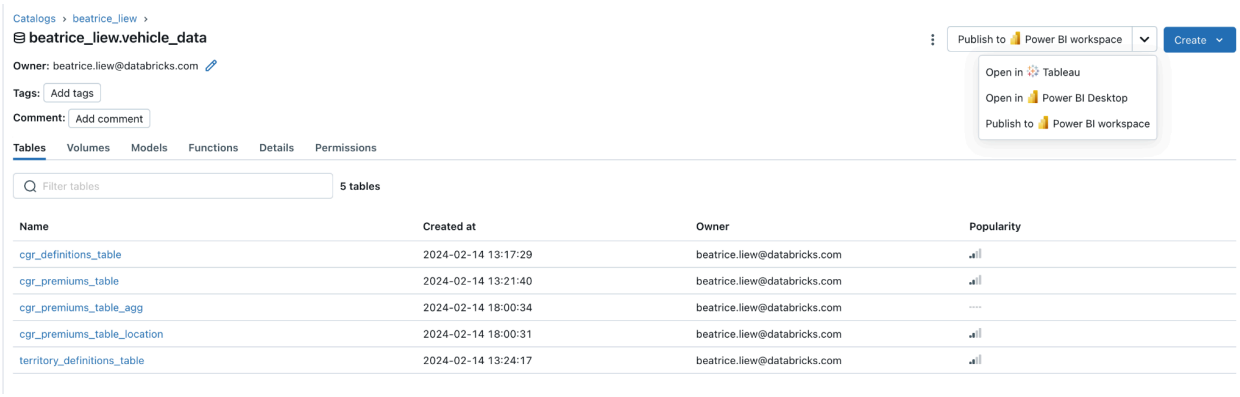
4. On the canvas, use the cgr_premiums_table_location dataset
5. To create a card with the total number of customers in the dataset, add a viz from the bottom blue bar. Under visualisation, select counter. Add count(*) as the value. Add in widget title to name it total number of customers
6. There's also an option to use text to vis, which allows you to use natural language to define the visual. Let's use what is the total value of current premium for another counter. Add in widget title to name it total premium value

7. To visualise the average underlying premium by county, use a bar visualisation. Select county on the x axis and underlying premium for the y axis (use AVG here). Add in avg for fixed expenses in the y axis to show the ratio of cost/premium
8. To visualise the relationship between the average cgr factor and the underlying premium, use a scatter plot. Select cgr_factor on the x axis and underlying_total_premium for the y axis (use AVG here).
9. Add a filter to allow us to filter the data with county. Notice that the other dataset will not be filtered as the column does not exist there.
10. Add another filter to allow us to filter the data on gender. As the column exists in both datasets, we will need to add both columns
11. This dashboard can be scheduled and shared

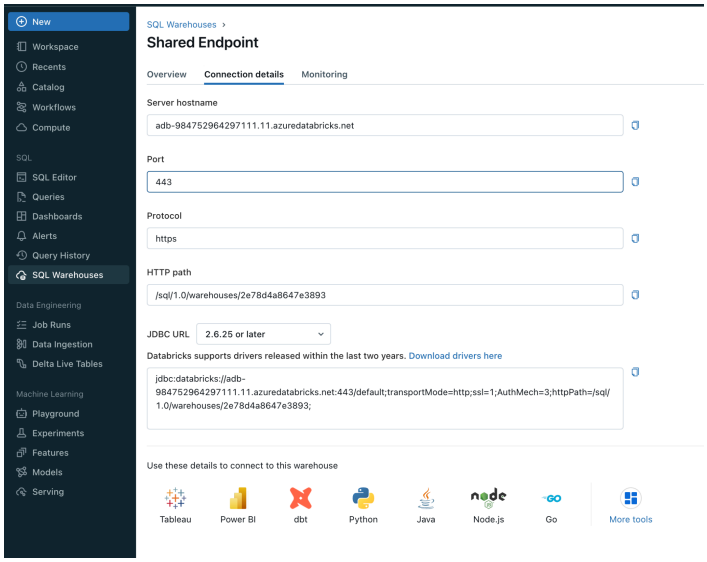


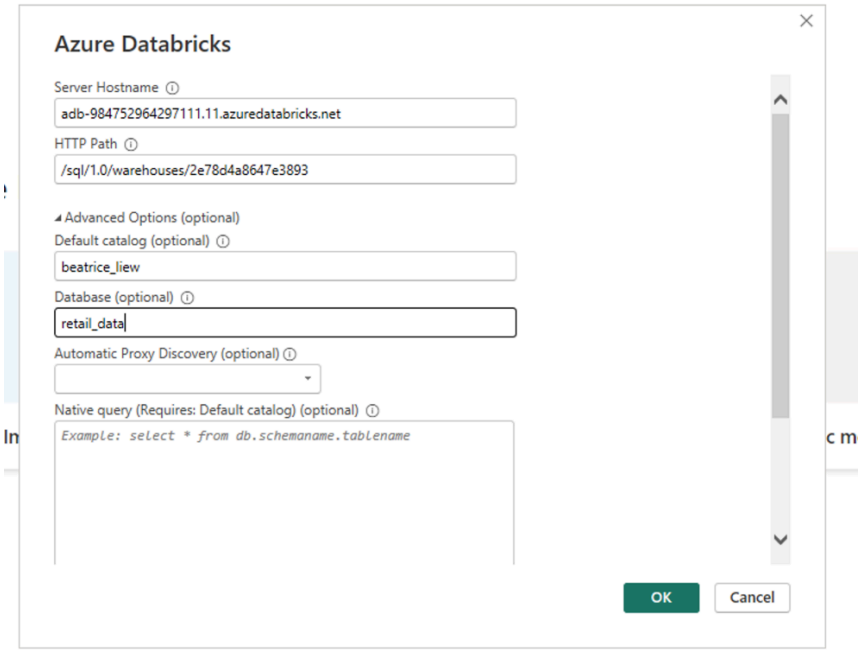
Connect to PBI

1. There are a few options to connect to PBI
    a. Click on open in PBI desktop. This will download a connection file that you could open directly on PBI desktop
    b. Click on publish to PBI service which can either publish the entire schema as a semantic model with relationships



c. Open PBI, click on Get data -> sign in under > Azure Databricks
    i. Fill out the information below, you can find the connection details on your DB SQL warehouse, under connection details
        1. Server hostname
        2. HTTP path
        3. Catalog (optional)
        4. Schema(optional)
    ii. Select the table to query





Genie demo

1. Using the Cgr_premiums_table table, click on create -> Genie Data Room
2. Within Data room, select all three tables used in this exercise and create room
3. Click on explain dataset to show how Genie explains the dataset from column names. Note that the explanation includes relationships mapped out during data modelling. But without modelling, data rooms can typically infer relationships if the col names are similar
4. Ask some example questions or use surprise me to get some insights from the data
5. Pick an example question and run it
    a. What is the age distribution of customers?
    b. Find the trend between average current_premium and the age of the customer

    c. Find the average current premium by town, which town has the highest premium?
    d. What is the average current premium for each gender across all territories?
    e. What is the average current premium for each gender split by county?
    f. Which quarters are customers born in? - Create and use get_quarter function
6. Ask some questions such as which age group has the highest current premium
7. Create a table function, log it to unity catalog and ask similar questions for Genie to trigger this function, which will be tagged as a trusted asset as end users can see that it's a curated answer that they can rely on

To create get_quarter function

```
create or replace function get_quarter(date DATE) RETURNS INT
RETURN
CASE WHEN MONTH(date) BETWEEN 2 AND 4 THEN 1
WHEN MONTH(date) BETWEEN 5 AND 7 THEN 2
WHEN MONTH(date) BETWEEN 8 AND 10 THEN 3
ELSE 4
END
```

To create a table function as a trusted asset:

```
CREATE OR REPLACE FUNCTION average_current_premium()
returns table (town string, average_current_premium double)
comment "What is the average current premium by town?"
return
SELECT
 t.town,
 AVG(p.current_premium) AS average_current_premium
FROM
 beatrice_liew.vehicle_data.cgr_premiums_table p
 JOIN beatrice_liew.vehicle_data.territory_definitions_table t ON p.territory =
t.territory
GROUP BY
 t.town
ORDER BY
 average_current_premium DESC;
```