Guide agent behavior with authored context for BigQuery data sources

This product or feature is subject to the "Pre-GA Offerings Terms" in the General Service Terms section of the Service Specific Terms (/terms/service-terms#1). Pre-GA products and features are available "as is" and might have limited support. For more information, see the launch stage descriptions (/products#product-launch-stages).

This page describes the recommended structure for writing effective prompts for your Conversational Analytics API (/gemini/docs/conversational-analytics-api/overview) data agents that connect to BigQuery data. These prompts are authored context that you define as strings by using the `system_instruction` parameter.

Examples of key components of system instructions

The following sections contain examples of key components of system instructions in BigQuery. These keys include the following:

- `tables` (#describe-your-data-with-tables)
- `fields` (#describe-fields)
- `measures` (#define-measures)
- `golden_queries` (#golden-queries)
- `golden_action_plans` (#golden-action-plans)
- `relationships` (#define-relationships)
- `glossaries` (#explain-glossaries)
- `additional_descriptions` (#additional-descriptions)

For descriptions of these key components, see the Guide agent behavior with authored context (/gemini/docs/conversational-analytics-api/data-agent-system-instructions) documentation page.

Describe your data with `tables`

The following YAML code block shows the basic structure for the `tables` key for the table `bigquery-public-data.thelook_ecommerce.orders`:

```
- tables:
    - table:
        - name: bigquery-public-data.thelook_ecommerce.orders
        - description: Data for customer orders in The Look fictitious e-commerc
        - synonyms:
            - sales
            - orders_data
        - tags:
            - ecommerce
            - transaction
```

Describe commonly used fields with `fields`

The following sample YAML code describes key fields such as `order_id`, `status`, `created_at`, `num_of_items`, and `earnings` for the `orders` table:

```
- tables:
    - table:
        - name: bigquery-public-data.thelook_ecommerce.orders
        - fields:
            - field:
                - name: order_id
                - description: The unique identifier for each customer order.
            - field:
                - name: user_id
                - description: The unique identifier for each customer.
            - field:
                - name: status
                - description: The current status of the order.
                - sample_values:
                    - complete
                    - shipped
                    - returned
            - field:
                - name: created_at
                - description: The timestamp when the order was created.
            - field:
                - name: num_of_items
                - description: The total number of items in the order.
                - aggregations:
                    - sum
                    - avg
            - field:
                - name: earnings
                - description: The sales amount for the order.
                - aggregations:
                    - sum
                    - avg
```

Define business metrics with `measures`

As an example, you can define a `profit` measure as a calculation of the earnings minus the cost as follows:

```
- tables:
    - table:
        - name: bigquery-public-data.thelook_ecommerce.orders
        - measures:
            - measure:
                - name: profit
                - description: Raw profit (earnings minus cost).
                - exp: earnings - cost
                - synonyms: gains
```

Improve accuracy with `golden_queries`

As an example, you can define golden queries for common analyses for the data in the `orders` table as follows:

```
- tables:
    - table:
        - golden_queries:
            - golden_query:
                - natural_language_query: How many orders are there?
                - sql_query: SELECT COUNT(*) FROM sqlgen-testing.thelook_ecommer
            - golden_query:
                - natural_language_query: How many orders were shipped?
                - sql_query: >-
                    SELECT COUNT(*) FROM sqlgen-testing.thelook_ecommerce.orders
                    WHERE status = 'shipped'
```

Outline multi-step tasks with `golden_action_plans`

As an example, you can define an action plan for showing order breakdowns by age group and include details about the SQL query and visualization-related steps:

```
- tables:
    - table:
        - golden_action_plans:
            - golden_action_plan:
                - natural_language_query: Show me the number of orders broken do
                - action_plan:
                    - step: >-
                        Run a SQL query that joins the table
                        sqlgen-testing.thelook_ecommerce.orders and
                        sqlgen-testing.thelook_ecommerce.users to get a
                        breakdown of order count by age group.
                    - step: >-
                        Create a vertical bar plot using the retrieved data,
                        with one bar per age group.
```

Define table joins with `relationships`

As an example, you can define an `orders_to_user` relationship between the `bigquery-public-data.thelook_ecommerce.orders` table and the `bigquery-public-`

`data.thelook_ecommerce.users` table as follows:

```
- relationships:
    - relationship:
        - name: orders_to_user
        - description: >-
            Connects customer order data to user information with the user_id an
        - relationship_type: many-to-one
        - join_type: left
        - left_table: bigquery-public-data.thelook_ecommerce.orders
        - right_table: bigquery-public-data.thelook_ecommerce.users
        - relationship_columns:
            - left_column: user_id
            - right_column: id
```

Explain business terms with `glossaries`

As an example, you can define terms like common business statuses and "OMPF" according to your specific business context as follows:

```
- glossaries:
    - glossary:
        - term: complete
        - description: Represents an order status where the order has been compl
        - synonyms: 'finish, done, fulfilled'
    - glossary:
        - term: shipped
        - description: Represents an order status where the order has been shipp
    - glossary:
        - term: returned
        - description: Represents an order status where the customer has returne
    - glossary:
        - term: OMPF
        - description: Order Management and Product Fulfillment
```

Include further instructions with `additional_descriptions`

As an example, you can use the `additional_descriptions` key to provide information about your organization as follows:

```
- additional_descriptions:
    - text: All the sales data pertains to The Look, a fictitious ecommerce stor
    - text: 'Orders can be of three categories: food, clothes, and electronics.'
```

Example: System instructions in BigQuery

The follow example shows sample system instructions for a fictitious sales analyst agent as follows:

```
- system_instruction: >-
    You are an expert sales analyst for a fictitious ecommerce store. You will a
- tables:
    - table:
        - name: bigquery-public-data.thelook_ecommerce.orders
        - description: Data for orders in The Look, a fictitious ecommerce store
        - synonyms: sales
        - tags: 'sale, order, sales_order'
        - fields:
            - field:
                - name: order_id
                - description: The unique identifier for each customer order.
            - field:
                - name: user_id
                - description: The unique identifier for each customer.
            - field:
                - name: status
                - description: The current status of the order.
                - sample_values:
                    - complete
                    - shipped
                    - returned
            - field:
                - name: created_at
                - description: >-
                    The date and time at which the order was created in timestam
                    format.
            - field:
                - name: returned_at
                - description: >-
                    The date and time at which the order was returned in timesta
                    format.
            - field:
                - name: num_of_items
                - description: The total number of items in the order.
                - aggregations: 'sum, avg'
            - field:
                - name: earnings
                - description: The sales revenue for the order.
                - aggregations: 'sum, avg'
            - field:
                - name: cost
                - description: The cost for the items in the order.
                - aggregations: 'sum, avg'
        - measures:
            - measure:
                - name: profit
                - description: Raw profit (earnings minus cost).
                - exp: earnings - cost
                - synonyms: gains
        - golden_queries:
            - golden_query:
                - natural_language_query: How many orders are there?
                - sql_query: SELECT COUNT(*) FROM sqlgen-testing.thelook_ecommer
            - golden_query:
                - natural_language_query: How many orders were shipped?
                - sql_query: >-
                    SELECT COUNT(*) FROM sqlgen-testing.thelook_ecommerce.orders
                    WHERE status = 'shipped'
        - golden_action_plans:
            - golden_action_plan:
                - natural_language_query: Show me the number of orders broken do
                - action_plan:
                    - step: >-
                        Run a SQL query that joins the table
                        sqlgen-testing.thelook_ecommerce.orders and
                        sqlgen-testing.thelook_ecommerce.users to get a
                        breakdown of order count by age group.
                    - step: >-
                        Create a vertical bar plot using the retrieved data,
                        with one bar per age group.
    - table:
        - name: bigquery-public-data.thelook_ecommerce.users
        - description: Data for users in The Look, a fictitious ecommerce store.
        - synonyms: customers
        - tags: 'user, customer, buyer'
        - fields:
            - field:
                - name: id
                - description: The unique identifier for each user.
            - field:
                - name: first_name
                - description: The first name of the user.
                - tag: person
                - sample_values: 'alex, izumi, nur'
            - field:
                - name: last_name
                - description: The first name of the user.
                - tag: person
                - sample_values: 'warmer, stilles, smith'
            - field:
                - name: age_group
                - description: The age demographic group of the user.
                - sample_values:
                    - 18-24
                    - 25-34
                    - 35-49
                    - 50+
            - field:
                - name: email
                - description: The email address of the user.
                - tag: contact
                - sample_values: '222larabrown@gmail.com, cloudysanfrancisco@gma
        - golden_queries:
            - golden_query:
                - natural_language_query: How many unique customers are there?
                - sql_query: >-
                    SELECT COUNT(DISTINCT id) FROM
                    bigquery-public-data.thelook_ecommerce.users
            - golden_query:
                - natural_language_query: How many users in the 25-34 age group
                - sql_query: >-
                    SELECT COUNT(DISTINCT id) FROM
                    bigquery-public-data.thelook_ecommerce.users WHERE users.age
                    '25-34' AND users.email LIKE '%@cymbalgroup.com';
    - relationships:
        - relationship:
            - name: orders_to_user
            - description: >-
                Connects customer order data to user information with the user_i
            - relationship_type: many-to-one
            - join_type: left
            - left_table: bigquery-public-data.thelook_ecommerce.orders
            - right_table: bigquery-public-data.thelook_ecommerce.users
            - relationship_columns:
                - left_column: user_id
                - right_column: id
- glossaries:
    - glossary:
        - term: complete
        - description: Represents an order status where the order has been compl
        - synonyms: 'finish, done, fulfilled'
    - glossary:
        - term: shipped
        - description: Represents an order status where the order has been shipp
    - glossary:
        - term: returned
        - description: Represents an order status where the customer has returne
    - glossary:
        - term: OMPF
        - description: Order Management and Product Fulfillment
- additional_descriptions:
    - text: All the sales data pertains to The Look, a fictitious ecommerce stor
    - text: 'Orders can be of three categories: food, clothes, and electronics.'
```

Last updated 2025-09-12 UTC.