**Problem Statement**

Spinners, realizing its expertise encompasses all things round, has, naturally,

decided to open a pizza delivery service! You will blend streaming and static data with a third party source to feed the data warehouse and finally power a reporting tool. The analytics team

would like to create a report based upon orders and the associated customer information. This

report will allow for a dropdown for the user to select a single US state, and it will display the

top 3 best-selling pizza combinations for that state along with:

* Total number of pizzas sold over the last 12 months for each type
* Gross sales (USD) over the last 12 months for each type
* Gross sales (USD) *per capita* over the last 12 months for each type
* Number of unique customers who ordered the item at least once over the last 12 months for each type

You are not responsible for the dashboarding tool but can assume it connects to any database

technology available (SQL, NoSQL, data warehouse, etc.). You are, however, responsible for

providing a single table, view, materialized view, etc. containing the relevant data such that it

can be fetched with only a filtering WHERE clause from the dashboard.

This dashboard should be low latency so that adjusting the state filter requires 5s or less for the

new data to populate over a stable, high-bandwidth internet connection.

**\*FOR SIMPLICITY, ASSUME A CUSTOMER CAN ONLY ORDER A SINGLE TYPE OF PIZZA PER**

**ORDER, BUT THEY CAN ORDER MULTIPLE UNITS OF THE SAME PIZZA IN ONE ORDER\***

**Data Sources**

You are provided two sources of data and should source a third yourself:

1. Transactional data that streams in over an API as customers place orders. Each order places an API request to an existing RESTful endpoint and provides a JSON payload with the following structure:

{

order\_id (integer): unique identifier for the order

customer\_id (integer): unique identifier for customer (foreign key to the customers

table)

type (string): "cheese", "pepperoni", "supreme", "meat lover", or "veggie" - the flavor of

pizza ordered

qty (integer): - the number of pizzas ordered

retail\_price (float): the total retail price of the order, including taxes

order\_date (timestamp in UTC): the time that the order was placed

}

1. A customer table in a RDBMS (PostgreSQL, MySQL, Oracle – whichever you prefer) containing the following columns:

* customer\_id (integer): primary key
* name (string): customer name
* address (string): street address
* city (string): city
* state (string): state
* zip\_code (string): five-digit US zip code

1. Adult population by US state. Please describe where you obtain this data and how you

will incorporate it into the pipeline.

**Requirements**

The transactional data is streaming in real time to the API, and the customer data is fixed in the

RDBMS. Assume no customers are added or removed over the lifetime of this pipeline for

simplicity. You must construct a cloud-based pipeline that takes each transactional API request

and joins it with the associated customer data, outputting the final record into a data

warehousing tool. This data warehouse should store all transactional records for all of history.

There is an SLO that every transaction placed should be available in the data warehouse **within**

**15 minutes of order placement**. This data warehouse table should feed into the data source

used for the reporting tool, which should contain the summary metrics described in the

Problem Statement above. Your solution should provide a means for a refresh of the reporting

data every day at 2:00am (if not more often). An ideal solution will keep the size of the data

(and hopefully therefore cost and latency) in the reporting database as small as possible.

**Challenge**

Assume Fabrikam has a total of 12 available development resources to work on this project,

consisting of Software Engineers, Data Engineers, QA Engineers, and Project Managers. You are

free to assemble a team using as few or as many of these as you see fit. The business would like

the project to be completed quickly but effectively so that resources are not tied up needlessly.

Please provide:

1. Your architectural solution to the above in as much detail as you see fit
2. A roadmap/workplan including a rough ETA of the work, the team required, workstreams/epics, and other relevant planning information

Choose any technologies or cloud provider that you are familiar with, but know that the more

specific you can be with which technologies you would choose, why you chose them, and how

you would use them is better.

Good luck!