

Technical Exercise

Overview	1
Analytical SQL Exercise	2
Objectives	2
Prerequisites	2
Data	2
Questions	3
Data Transformation Exercise	4
Objectives	4
Prerequisites	4
Familiarize yourself with dbt	4
Set up your environment	5
Start Exercise	6
Develop data models addressing business questions:	7
Your Resources: Two tables	7
"TE_RAW"."SCOOTERS"."SCOOTER_LOCATIONS"	7
"TE_RAW"."SCOOTERS"."US_HEX7_MSA"	8
Exercise Requirements	8

Overview

This exercise is composed of two sections, Analytical SQL Exercise and Data Transformation exercise.

Although the exercises use related data, they are independent of each other and can be completed separately.

Repository Setup: 205 Data Lab will create a single private GitHub repository for both exercises. The repository will contain a [SUBMISSION.md](#) template file where you will document your exercise answers. Your dbt code will also be committed to this same repository.

Analytical SQL Exercise

Objectives

The objective of this section is to use SQL to answer business questions below.

Prerequisites

205 Data Lab will provide access to the company's Snowflake account:

<https://dz46997.us-east-1.snowflakecomputing.com/>

Your credentials to access the account are to be provided separately by your 205 Data Lab contact.

Data

The Snowflake account contains a database with data to be used in the Analytical SQL Exercise. The tables are:

- `"TE_ANALYTICS"."BI"."SCOOTERS_PARSED"`: This table contains GPS (latitude and longitude readings) for multiple electric scooters over time.
 - It also contains a hex7 column that maps the latitude and longitude to a geographical index (an ID that represents a geographical region on the earth's surface with an approximate area of 5 square kms.). Although this exercise does not require you to understand more about the indexes and how they are computed, if you are curious, you can learn more about Uber's H3 hexagonal grid system [here](#).
 - You can use the `decoded_id` as the scooter identifier.
 - The `extracted_at` field represents the time at which the scooter reported its position. For date-based aggregations, use the date portion of this timestamp field.
- `"TE_ANALYTICS"."BI"."US_HEX7_MSA"`: This table maps the hex7 index to an MSA (metropolitan statistical area). We use this table to know in what MSA each scooter is located at a given point in time. This table contains the Hex7 identifiers and the matching MSAs you will need to obtain for identifying the MSA associated with each scooter reading. MSA stands for Metropolitan Statistical Area. It's a geographical region

that has one or more US cities. Note that some scooters may be outside of the United States. The MSA concept only applies to the United States and the scope of this exercise is the United States only.

Note: The SQL Exercise uses pre-processed tables in the `TE_ANALYTICS` schema for convenience. The DBT Exercise will work with raw data in the `TE_RAW` schema, where you'll build transformations to create similar analytical tables.

Questions

Please write queries to answer the following questions:

SQL1) Write a query that computes for each MSA, the unique scooter count, and unique scooter count as a % of all scooters, displaying only the top 3 MSAs (by scooter count) for the month of January 2020. Provide a table with your results.

SQL2) Write a query that computes the overall rank of MSAs based on the number of unique scooters for March 3rd, 2020. The MSA with the most scooters will have rank = 1.

SQL3) Provide a 100% stacked column chart showing the % scooter share of each provider company for each day of 2020 for the MSA that contains Washington, DC. Each column represents one day, with provider shares stacked to total 100%. The scooter share is the number of unique scooters a provider has in an MSA on a given day as a percentage of all the scooters seen that day in the same MSA. (Please feel free to use spreadsheet of your choice, e.g. google sheets or ms excel , or other. Feel free to use a visualization tool if available.)

Submitting Your SQL Answers:

1. Locate the `SUBMISSION.md` file in your GitHub repository (you'll receive repository access from your 205 Data Lab contact)
2. Complete Part 1 (Analytical SQL Exercise) of the `SUBMISSION.md` file with:
 - Your SQL queries (in code blocks)
 - Query results (as markdown tables)
 - Your visualization for SQL3
3. Commit and push your updated `SUBMISSION.md` to the repository

TIP: Generate markdown tables easily by copy-pasting your Snowflake results into this online tool: https://www.tablesgenerator.com/markdown_tables

Data Transformation Exercise

Objectives

The objective of this exercise is to give you an opportunity to apply some of the tools and approaches our team uses regularly in real-world projects:

- Use git and GitHub to collaborate on a project
- Use SQL to transform and analyze data
- Quickly learning a new technology (dbt)
- Solve problems with limited hand-holding
- Understand requirements based on written specifications or instructions
- Produce structured and well-designed code
- Design and implement a data warehouse and model data for analytical/business use

Prerequisites

This exercise requires that you:

- Share your GitHub username with 205 Data Lab
- Have installed git in your CLI environment

205 Data Lab will:

- Provide access to the company's Snowflake instance. The Snowflake account contains a database with raw data to be used in a Data Transformation Exercise. We'll also give you access to the database (**TE_ANALYTICS**) where the output tables/views will be created.
- Create a private GitHub repository and share it with you. This is the same repository containing the `SUBMISSION.md` file from the SQL Exercise. You will push your dbt code to this repository. The repository will be named as follows:
`205datalab-ext/te_yyyymmdd_<first name initial><last name>`

Familiarize yourself with dbt

One of the objectives of this exercise is quickly learning a new technology - "dbt". Dbt is used by 205 Data Lab extensively. It's a great tool to build SQL-based transformations that leverage the power of the data warehouse. There is an excellent course at this link: <https://courses.getdbt.com/courses/fundamentals>. The key sections of this course relevant to this exercise are:

- Welcome to dbt Fundamentals
- Who is an analytics engineer?
- Set up dbt Cloud
- Sources
- Models
- Tests

- Project Documentation

PLEASE NOTE: While the course is several hours long, you don't need to have completed it to finish the activity below. You can also take advantage of the various sections on <https://getdbt.com/>. Additionally, the dbt documentation is very good and you'll find references to the contents above in written form: [dbt documentation](#).

NOTE: You can skip the 'Set up dbt Cloud' section since this exercise uses dbt Core (the dbt CLI).

Set up your environment

- **Install dbt on your machine** (Please note: This exercise requires the dbt CLI, not dbt cloud)
Installation instructions: <https://docs.getdbt.com/dbt-cli/installation>
Make sure you run `dbt --version` to make sure that the installation is completed successfully.
- **Clone the shared git repository locally on your machine at your desired directory.**
You will see that the repository will be created under the following name: `te_yyyymmdd_<first name initial><last name>`. Review the contents.
Please see the `profiles_temp.yml` file has been provided in the repository. You will use this file after the next step.
- **Set up your profile** There is a sample `profiles_temp.yml` file in the repo. The user needs to update it with their username and password,, rename as `profiles.yml` and **move** to your home directory: `~/dbt/profiles.yml`.
This file tells dbt how to connect to the database and where to create the tables/views.
Important: Do not commit your `profiles.yml` file to the repository as it contains credentials. The `.gitignore` file in the repo is configured to exclude this file.
- **Test and confirm that your environment is set up correctly by running `dbt run`.**
This should create some sample models.

```
> dbt run
Running with dbt=0.19.2
Found 2 models, 4 tests, 0 snapshots, 0 analyses, 143 macros, 0 operations, 0 seed files, 0 sources, 0
12:19:34 | Concurrency: 4 threads (target='dev')
12:19:34 |
12:19:34 | 1 of 2 START table model te_user1.my_first_dbt_model..... [RUN]
12:19:36 | 1 of 2 OK created table model te_user1.my_first_dbt_model..... [SUCCESS 1 in 2.38s]
12:19:36 | 2 of 2 START view model te_user1.my_second_dbt_model..... [RUN]
12:19:38 | 2 of 2 OK created view model te_user1.my_second_dbt_model..... [SUCCESS 1 in 1.42s]
12:19:39 |
12:19:39 | Finished running 1 table model, 1 view model in 8.27s.

Completed successfully

Done. PASS=2 WARN=0 ERROR=0 SKIP=0 TOTAL=2
```

- **Make sure you can access the snowflake instance and explore the source tables at**
<https://dz46997.us-east-1.snowflakecomputing.com/>
- **Make sure you have been provided by 205 Data Lab with your Snowflake account credentials.**

Start Exercise

Client: Your hypothetical client in this exercise is a scooter rental company.

Objective: Your job is to write a data transformation pipeline using SQL, orchestrated by dbt. You will create a few tables (data models) that can be used to answer business questions. You will implement the transformation steps; and create the schema tests and documentation of your data transformation pipeline.

Develop data models addressing business questions:

As a deliverable of this exercise, you need to develop data model(s) where the final output shows the answer to the following question as a model (i.e. a “select * “ statement from a final model will give the answer to one of the following questions :

1. How many unique scooters each provider has in each MSA on any given day? (Please name your model `fct_provider_msa_day.sql`)
2. What is the monthly average of daily scooter displacement? Displacement is defined as the distance between the first position of each scooter on a given day and the last position of the scooter that same day. Your final model should be able to provide this metric by company and MSA. Hint: the database has a function that calculates the distance between two latitude/longitude pairs. (Please name your final model `fct_company_msa_avg_disp_monthly.sql`)

Your Resources: Two tables

"TE_RAW"."SCOOTERS"."US_HEX7_MSA (Ready to be used)

"TE_RAW"."SCOOTERS"."SCOOTER_LOCATIONS (Needs to be transformed)

"TE_RAW"."SCOOTERS"."SCOOTER_LOCATIONS"

The SCOOTER_LOCATIONS table has semi-structured data (originally json) that was loaded into the database as a variant data type.

Each row represents the position of a scooter at a given point in time. You can use the `decoded_id` as the scooter identifier. The `extracted_at` field represents the time at which the scooter reported its position. The provider slug is the name of the company renting the scooter.

Example row:

```
{
  "attributes": [
    "ELECTRIC"
  ],
  "battery": 67,
  "decoded_id": "voi:SCOOTER:0596208e-add2-4e9c-9073-dfdf677abc99",
  "extracted_at": "2019-12-01T04:40:08.508564",
  "hexagons": {
    "10": "8a1fa0a0402ffff",
    "6": "861fa0a07ffffff",
    "7": "871fa0a04ffffff",
    "8": "881fa0a041ffffff",
    "9": "891fa0a0403ffff"
  },
  "id":
  "dm9p0lNDT09URVI6MDU5NjIwOGUtYWRkMi00ZTljLTkwNzMtZGZkZjY3N2FiYzk5",
  "lat": 50.78412628173828,
  "lng": 6.097358226776123,
  "provider": {
    "slug": "voi"
  },
  "type": "SCOOTER",
  "voiFields": {
    "bounty": 0,
    "locked": true,
    "name": "VOI",
    "short": "5n7h",
    "status": "ready",
    "type": "como",
    "zone": 176
  }
}
```

Extracting Hex7 for MSA Mapping: The `hexagons` object contains H3 indexes at various resolutions (6, 7, 8, 9, 10). To join with the `US_HEX7_MSA` table, you'll need to extract the hex7 index, which is stored under `hexagons.7`.

For date-based aggregations, extract the date portion from the `extracted_at` timestamp field. You may assume all timestamps are in UTC.

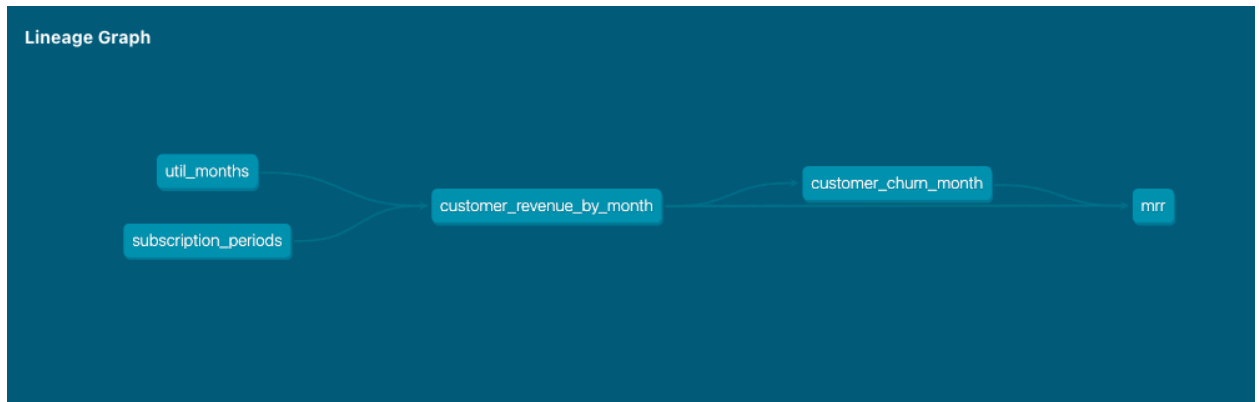
```
"TE_RAW"."SCOOTERS"."US_HEX7_MSA"
```

This table maps the hex7 index (hexagon of resolution 7) to an MSA (metropolitan statistical area). We use this table to know in what MSA each scooter is located at a given point in time. This table contains the Hex7 identifiers and the matching MSAs you will need to obtain for identifying the MSA associated with each scooter reading. MSA stands for Metropolitan Statistical Area. It's a geographical region that has one or more US cities. Note that some scooters may be outside of the United States. The MSA concept only applies to the United States and the scope of this exercise is the United States only.

Exercise Requirements

- Start by creating two `dbt sources` for each of the two tables above. (Please see [dbt documentation](#) for “sources”).
- Create a “staging” model for each source in a “staging” directory under the “models” directory. One and only one staging model should access each source. A “staging” model is a model that serves as the only entry point to the raw data. Please see the [“Best Practices”](#) section of the dbt documentation.
Please note that the raw scooter data is in semi-structured format so you will need to extract the required elements using Snowflake’s semi-structured data operators. Do this in the staging model so you don’t have to extract the fields repeatedly in dependent models. You may want to refer to [this link](#) on Snowflake documentation on how to extract semi-structured data.
- After the `staging` model, introduce a model that builds a deduplication step, where a single scooter has a single reading at a certain time. Make sure the name of this model contains the “tmp_” prefix. It is expected that you’ll use a window function to dedup the data. Place this model in a `tmp` directory in the models directory.
- Please introduce other `tmp_` models as needed
- Finally, the model containing your final answer to the business question should be named with the `fact_` prefix (as named in the question). Please place this model in the `marts` directory.
- Use Common Table Expressions (CTEs) instead of subqueries for readability
- Make sure you use dbt model references but not explicit Snowflake table names in your queries. Using references is how dbt knows the relationship between models in your project
- Every model should have a primary key. If there is no primary key in the raw data, create a composite primary key combining two or more columns so that each row can be uniquely identified.
- Add “Unique” and “not_null” dbt “schema” tests for the primary keys. You can place these in a `schema.yml` file in the same folder where you put your model.
- Run `dbt build` after finalizing each of your models, and at the end of the project and make sure your dbt project does not return errors.

- To make sure your code works as expected, check the resulting tables in the Snowflake UI.
- Commit your code and push to the GitHub repository.
- When finished, generate documentation, `dbt docs generate`, and then `dbt docs serve` view the results on your browser, capture a screenshot, and upload the screenshot to the repo. Below is an example of a dbt docs screenshot.



Final Submission Checklist:

Before notifying your 205 Data Lab contact that you've completed the exercise, ensure:

- `SUBMISSION.md` is fully completed with all SQL queries, results, and visualization
- All dbt models are committed and pushed to the repository
- dbt documentation screenshot is in the repository
- `dbt build` runs without errors
- No credentials are committed to the repository