Yelp Data Python ETL & SQL Analytics Project

Author: Aiden Zhu

Tech Stack: Python (Pandas, JSON), Kaggle Notebook, SQLAlchemy, Azure SQL Database, Azure Data Studio

Abstract

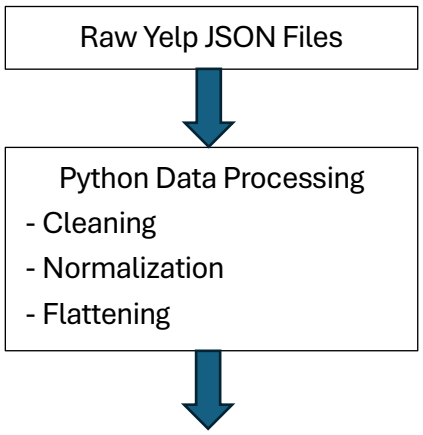
This project demonstrates an end-to-end ETL pipeline that transforms raw nested Yelp JSON files into a clean, relational Azure SQL database optimized for analytical queries.
The pipeline covers JSON data ingestion, cleaning, normalization, schema design, relational modeling, primary/foreign key configuration, and the execution of SQL analyses to generate actionable business insights.
The goal of the project is to illustrate my ability to work with complex semi-structured data, convert it into a usable relational schema, and apply analytical SQL to extract meaningful insights—skills directly relevant to data analyst and data engineering roles.

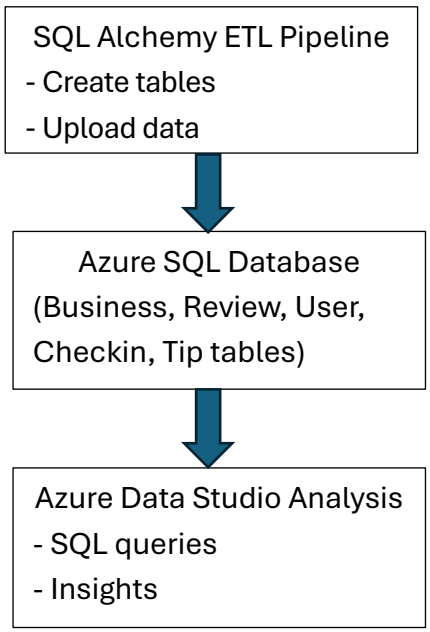
Project Overview

The Yelp dataset contains multiple nested JSON files, including business information, reviews, users, tips, and check-ins. These files are highly unstructured, containing irregular data types, null values, and nested objects.
To convert the raw JSON dataset into a usable analytics database, I executed the following:

- Loaded, inspected, and cleaned JSON data using Python
- Normalized nested attributes using `json_normalize` and custom parsing
- Standardized data types (e.g., converting IDs to varchar(22))
- Flattened multi-value fields such as categories and hours
- Designed a relational schema for Azure SQL
- Created tables with PK/FK constraints
- Uploaded data using SQLAlchemy
- Validated row/column counts in Azure Data Studio
- Conducted SQL-based business analysis (top categories, review distribution, rating insights)

Architecture Diagram





ETL Workflow Summary

Step 1 — JSON Loading

Each JSON file was read into Python using buffered loading to avoid memory overload.

Step 2 — Data Type Standardization

Several fields used inconsistent types (e.g., `varchar(max)`).

Converted IDs to `varchar(22)` to ensure key consistency.

Step 3 — Flattening Nested Structures

Tools used:

- `json_normalize()`
- `.split(',')` for multi-category fields
- Dictionary expansion for attributes/hours

Step 4 — Upload to Azure

Tables were created using SQLAlchemy:

- Defined column types
- Uploaded DataFrame to SQL tables
- Validated insertion counts

Step 5 — Relational Modeling

Primary and foreign keys were added in Azure Data Studio.

JSON Normalization & Cleaning (Text Version + Code Snippets)

Handling Nested Attributes

For example:

- `attributes` → expanded into separate columns
- `categories` → split into lists
- `hours` → converted from dictionary to structured fields

```
file_business = open("/kaggle/input/yelp-dataset/yelp_academic_dataset_business.json")
business = []
for line in file_business:
    business.append(json.loads(line))

business_df = pd.json_normalize(business)

business_df['categories'] = business_df['categories'].str.split(", ")

#business_df.to_excel("business_data.xlsx", index=False)
print(business_df.head())
file_business.close()
```

Figure 1: JSON Normalization Example – Business File

Azure SQL Schema & ERD Diagram

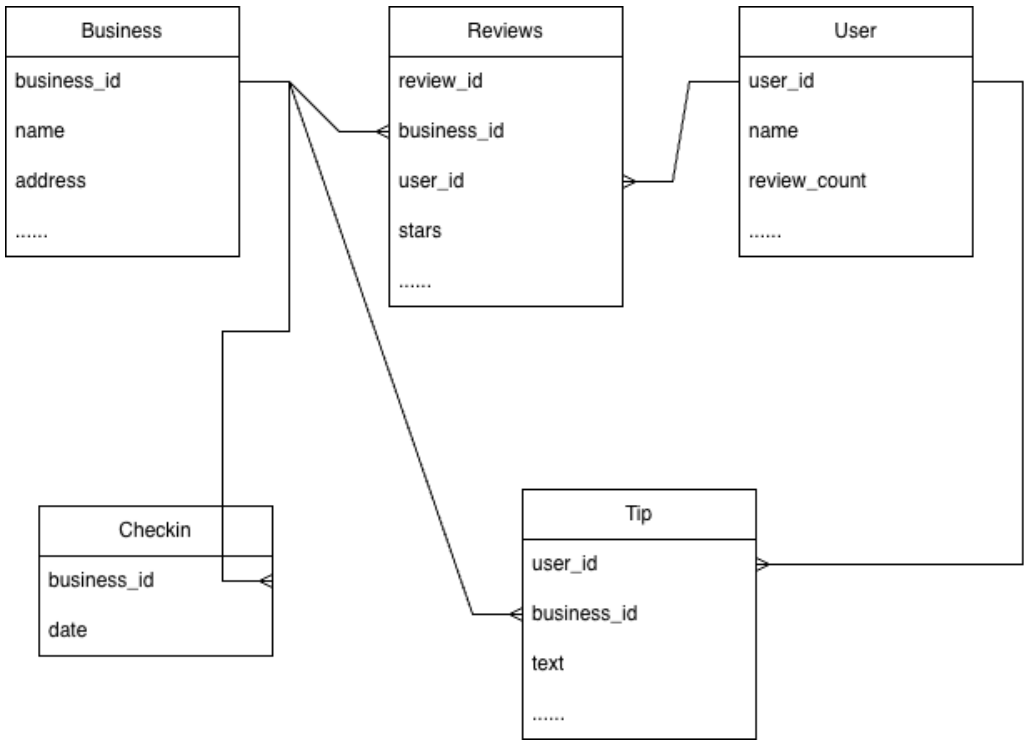


Figure 2: ERD Diagram of Yelp Data

Key Analytical SQL Queries & Insights Examples

Insight 1 — Most Reviewed Restaurant Categories

Mexican restaurants lead with the highest review count (6,231 reviews), showing strong customer engagement.

```
%%sql
SELECT TOP 10 categories, COUNT(*) AS review_count
FROM review_df
JOIN business_df ON review_df.business_id = business_df.business_id
WHERE UPPER(business_df.categories) LIKE ('%RESTAURANT%') AND city = 'Tucson'
GROUP BY categories
ORDER BY review_count DESC;
```

```
 * mssql+pyodbc://sqladmin:***@zienzhu2-server.database.windows.net:1433/mini-project-db?driver=ODBC+Driver+17+for+SQL+Server
Done.
```

categories	review_count
Restaurants, Mexican	6258
Mexican, Restaurants	5913
Restaurants, Cocktail Bars, Bars, Comfort Food, American (New), Sandwiches, Nightlife, Breakfast & Brunch	2230
Restaurants, American (New)	2078
Tapas/Small Plates, Desserts, Cocktail Bars, Nightlife, Bars, Tacos, Mexican, Food, Restaurants, Salad	1617
Chinese, Restaurants	1349
Restaurants, Chinese	1340
American (Traditional), Restaurants, Mexican, Latin American	1331
Ice Cream & Frozen Yogurt, American (New), Nightlife, Sandwiches, Bars, Restaurants, Salad, American (Traditional), Desserts, Food	1329
Italian, Restaurants	1327

Figure 3: Query 4 – Reviews of Different Type of Restaurants

Insight 2 — Review Sentiment Indicators

Sentiment Analysis: Differences Between High-Rated (≥4) and Low-Rated (≤2) Reviews:

- **High-rated reviews** tend to include more **positive and affirmative expressions**, such as *"I've"*, *"definitely"*, and other words that convey certainty, satisfaction, or strong approval. (Figure 4)
- **Low-rated reviews** often contain more **negative contractions and time-related terms**, such as *"don't"*, *"didn't"*, or mentions of waiting time like *"minutes"*, which are typically used to express complaints or dissatisfaction. (Figure 5)

```
%%sql

SELECT TOP 20 word, COUNT(*) AS frequency
FROM (
        SELECT value AS word, review_df.business_id
        FROM review_df
        CROSS APPLY STRING_SPLIT(text, ' ')    -- This splits the 'text' column into words
        WHERE stars >= 4
) AS positive_words
JOIN business_df ON positive_words.business_id = business_df.business_id
WHERE UPPER(business_df.categories) LIKE '%RESTAURANT%'
AND city = 'Tucson'
AND UPPER(business_df.categories) LIKE '%MEXICAN%'
AND upper(word) NOT IN ('  ','I','THE','AND','A','TO','IS','WAS','OF','IN','BUT','ON','FOR','IT','W
'YOU','THEY','ARE','HAD','MY','IF','AN','HAVE','WERE','SO','GOOD','GREAT','VERY','NOT','AS','BE','AT','
'OUR','HERE','LIKE','BEST','ONE','JUST','FROM','WILL','GET','THERE','REALLY','OUT','GO','LOVE','ALWAYS'
,'YOUR','WOULD','SHOULD','MIGHT','HATE','POOR','OR','HAS','ABOUT','CAN','BACK','NO','SHE','HE','US','GO
'BY','THEN','SAID','WENT','OTHER','IT_S','%DON''T%','DO NOT','%DID''NT%','ASKED','MORE','ASKED','TOLD'
'-','TRY','LITTLE')
GROUP BY word
ORDER BY frequency DESC;
```

```
 * mssql+pyodbc://sqladmin:***@zienzhu2-server.database.windows.net:1433/mini-project-db?drive
Done.
```

word	frequency
food	20086
place	13818
Mexican	9544
it's	7796
salsa	7760
service	7607
tacos	7133
I've	5256
definitely	5222
restaurant	5093
Tucson	4926
ordered	4867

Figure 4: Query 5 – Words Frequency of High Rating Reviews

```
%%sql

SELECT TOP 20 word, COUNT(*) AS frequency
FROM (
        SELECT value AS word, review_df.business_id
        FROM review_df
        CROSS APPLY STRING_SPLIT(text, ' ')    -- This splits the 'text' column into words
        WHERE stars < 3
) AS negative_words
JOIN business_df ON negative_words.business_id = business_df.business_id
WHERE UPPER(business_df.categories) LIKE '%RESTAURANT%'
AND city = 'Tucson'
AND UPPER(business_df.categories) LIKE '%MEXICAN%'
AND upper(word) NOT IN (' ','I','THE','AND','A','TO','IS','WAS','OF','IN','BUT','ON','FOR','IT','WITH','TH
'YOU','THEY','ARE','HAD','MY','IF','AN','HAVE','WERE','SO','GOOD','GREAT','VERY','NOT','AS','BE','AT','THEIR','
'OUR','HERE','LIKE','BEST','ONE','JUST','FROM','WILL','GET','THERE','REALLY','OUT','GO','LOVE','ALWAYS','ALSO',
,'YOUR','WOULD','SHOULD','MIGHT','HATE','POOR','OR','HAS','ABOUT','CAN','BACK','NO','SHE','HE','US','GOT','EVEN
'BY','THEN','SAID','WENT','OTHER','IT_S','%DON''T%','DO NOT','%DID''NT%','ASKED','MORE','ASKED','TOLD','MENTIO
GROUP BY word
ORDER BY frequency DESC;
```

```
 * mssql+pyodbc://sqladmin:***@zienzhu2-server.database.windows.net:1433/mini-project-db?driver=ODBC+
Done.
```

word	frequency
food	9339
place	4510
ordered	3700
service	3680
order	3121
Mexican	2765
Don't	2642
didn't	2537
restaurant	2354
It's	2176
minutes	2137
salsa	2046
tacos	1810
two	1784
do	1783
taco	1755
going	1746

Figure 5: Query 6 – Words Frequency of Low Rating Reviews

Insight 3 — Weekly Checking Patterns

- **Customer traffic peaks on weekends**, especially **Saturday** (117 check-ins).
- **Monday and Tuesday show the lowest demand**, with only **48–60** check-ins.

Business Action:

To optimize labor cost and operational efficiency, the restaurant should **increase staffing on weekends** and **reduce staffing on Mondays and Tuesdays**, when customer volume is minimal.

```
%%sql
SELECT
        CASE
                WHEN DATEPART(WEEKDAY, CONVERT(DATETIME,
                        SUBSTRING(checkin_df.date, 1, CHARINDEX(',', checkin_df.date + ',') - 1))) = 1 THEN 'Sunday'
                WHEN DATEPART(WEEKDAY, CONVERT(DATETIME,
                        SUBSTRING(checkin_df.date, 1, CHARINDEX(',', checkin_df.date + ',') - 1))) = 2 THEN 'Monday'
                WHEN DATEPART(WEEKDAY, CONVERT(DATETIME,
                        SUBSTRING(checkin_df.date, 1, CHARINDEX(',', checkin_df.date + ',') - 1))) = 3 THEN 'Tuesday'
                WHEN DATEPART(WEEKDAY, CONVERT(DATETIME,
                        SUBSTRING(checkin_df.date, 1, CHARINDEX(',', checkin_df.date + ',') - 1))) = 4 THEN 'Wednesday'
                WHEN DATEPART(WEEKDAY, CONVERT(DATETIME,
                        SUBSTRING(checkin_df.date, 1, CHARINDEX(',', checkin_df.date + ',') - 1))) = 5 THEN 'Thursday'
                WHEN DATEPART(WEEKDAY, CONVERT(DATETIME,
                        SUBSTRING(checkin_df.date, 1, CHARINDEX(',', checkin_df.date + ',') - 1))) = 6 THEN 'Friday'
                WHEN DATEPART(WEEKDAY, CONVERT(DATETIME,
                        SUBSTRING(checkin_df.date, 1, CHARINDEX(',', checkin_df.date + ',') - 1))) = 7 THEN 'Saturday'
        END AS weekday_of_week,
        COUNT(*) AS checkin_count
FROM
        checkin_df
JOIN
        business_df ON checkin_df.business_id = business_df.business_id
WHERE
        business_df.categories LIKE '%RESTAURANT%'
        AND city = 'Tucson'
        AND business_df.categories LIKE '%MEXICAN%'
GROUP BY
        DATEPART(WEEKDAY, CONVERT(DATETIME,
                SUBSTRING(checkin_df.date, 1, CHARINDEX(',', checkin_df.date + ',') - 1)))
ORDER BY
        checkin_count DESC;
```

```
 * mssql+pyodbc://sqladmin:***@zienzhu2-server.database.windows.net:1433/mini-project-db?driver=ODBC+Driver+17+for+SQL+
Done.
```

weekday_of_week	checkin_count
Saturday	117
Friday	84
Sunday	76
Tuesday	64
Wednesday	60
Thursday	57
Monday	48

Figure 6: Query 7 – Weekly Checking Patterns

Conclusions

This project showcases the ability to:

- Transform semi-structured data into a relational data warehouse
- Utilize Python to clean and normalize nested JSON
- Design effective SQL schemas with PK/FK
- Utilize Azure Cloud services
- Conduct analytical SQL queries to reveal business insights

These skills directly apply to real-world data engineering and analytics workflows.

Appendix

- Azure server creation screenshots

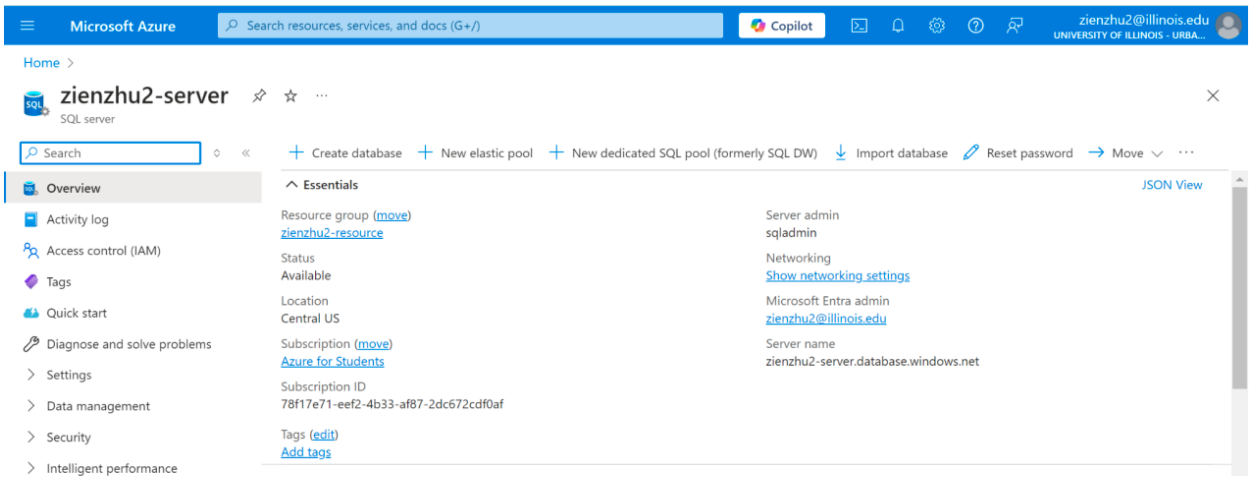


Figure 7: Server Overview

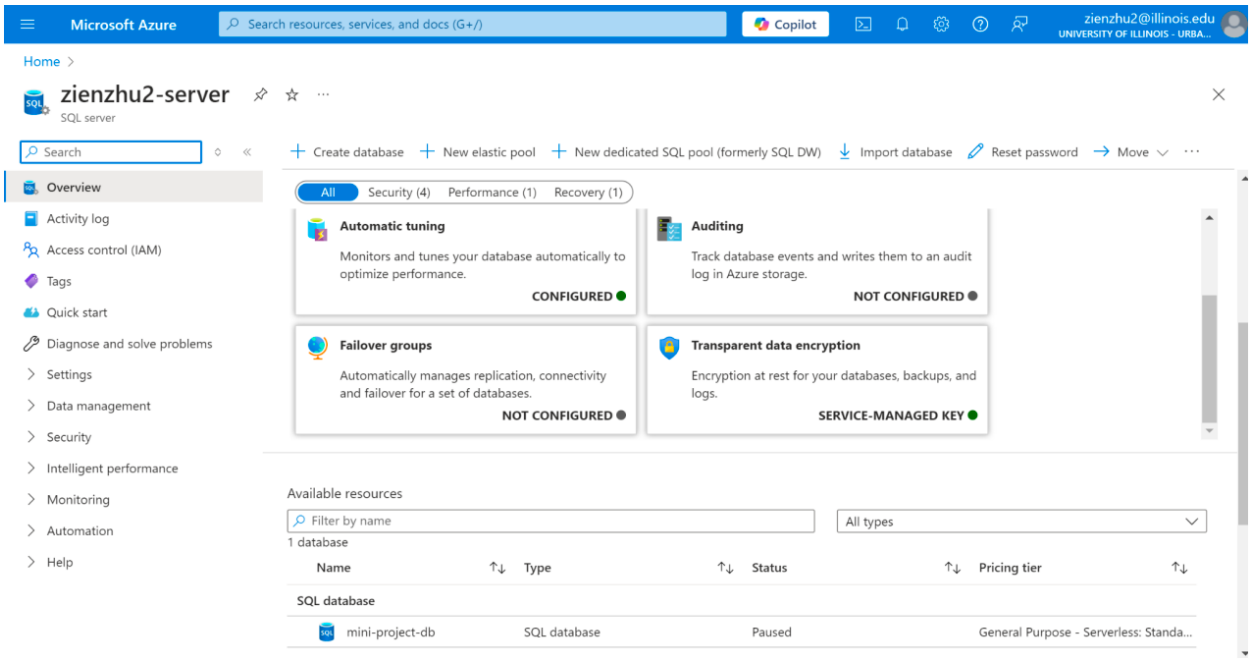


Figure 8: Database Creation

- Python normalization code

```
file_business = open("/kaggle/input/yelp-dataset/yelp_academic_dataset_business.json")
business = []
for line in file_business:
    business.append(json.loads(line))

business_df = pd.json_normalize(business)

business_df['categories'] = business_df['categories'].str.split(", ")

#business_df.to_excel("business_data.xlsx", index=False)
print(business_df.head())
file_business.close()
```

```
              business_id                      name  \
0  Pns2l4eNsfO8kk83dixA6A  Abby Rappoport, LAC, CMQ
1  mpf3x-BjTdTEA3yCZrAYPw             The UPS Store
2  tUFrWirKiKi_TAnsVWINQQ                    Target
3  MTSW4McQd7CbVtyjqoe9mw         St Honore Pastries
4  mWMc6_wTdE0EUBKIGXDVfA  Perkiomen Valley Brewery
```

Figure 9: Python Normalization – Business

```
file_tip = open("/kaggle/input/yelp-dataset/yelp_academic_dataset_tip.json")
tip = []
for line in file_tip:
    tip.append(json.loads(line))

tip_df = pd.json_normalize(tip)
print(tip_df.head())
#tip_df.to_csv("tip_data.csv",index=False)
file_tip.close()
```

```
                  user_id             business_id  \
0  AGNUgVwnZUey3gcPCJ76iw  3uLgwr0qeCNMjKenHJwPGQ
1  NBN4MqHP9D3cw--SnauTkA  QoezRbYQncpRqyrLH6Iqjq
```

Figure 10: Python Normalization – Tip

```python
file_checkin = open("/kaggle/input/yelp-dataset/yelp_academic_dataset_checkin.json")
checkin = []
for line in file_checkin:
    checkin.append(json.loads(line))

records = []
for entry in checkin:
    business_id = entry['business_id']
    date = entry['date'].split(", ")

    for time in date:
        records.append({
            'business_id': business_id,
            'date': time
        })
checkin_df = pd.DataFrame(records)
print(checkin_df.head())
#checkin_df.to_csv("checkin_data.csv",index=False)
file_checkin.close()
```

```
              business_id                 date
0  ---kPU91CF4Lq2-WlRu9Lw  2020-03-13 21:10:56
1  ---kPU91CF4Lq2-WlRu9Lw  2020-06-02 22:18:06
2  ---kPU91CF4Lq2-WlRu9Lw  2020-07-24 22:42:27
3  ---kPU91CF4Lq2-WlRu9Lw  2020-10-24 21:36:13
```

Figure 11: Python Normalization - Checkin

```python
file_review = open("/kaggle/input/yelp-dataset/yelp_academic_dataset_review.json")

chunk_size = 10000
review_chunks = []
review_df = pd.DataFrame()

for i, line in enumerate(file_review):
    try:
        review_data = json.loads(line)
        review_chunks.append(review_data)

        if (i + 1) % chunk_size == 0:
            chunk_df = pd.json_normalize(review_chunks)
            review_chunks = []

            review_df = pd.concat([review_df, chunk_df], ignore_index=True)

    except json.JSONDecodeError as e:
        print(f"JSON decode error at line {i}: {e}")

if review_chunks:
    chunk_df = pd.json_normalize(review_chunks)
    review_df = pd.concat([review_df, chunk_df], ignore_index=True)

print(review_df.head())
#review_df.to_csv("review_data.csv",index=False)
file_review.close()
```

```
                review_id                 user_id             business_id  \
0  KU_O5udG6zpxOg-VcAEodg  mh_-eMZ6K5RLWhZvISBhwA  XQfwVwDr-v0ZS3_CbbE5Xw
```

Figure 12: Python Normalization – Review

- SQLAlchemy upload logs

```
from sqlalchemy import create_engine
import sqlalchemy
!pip install pyodbc
!pip install sqlalchemy
!apt-get install -y unixodbc-dev

!curl https://packages.microsoft.com/keys/microsoft.asc | apt-key add -
!curl https://packages.microsoft.com/config/ubuntu/$(lsb_release -rs)/prod.list > /etc/apt/sources.list.d/mssql-release.list
!apt-get update
!ACCEPT_EULA=Y apt-get install -y msodbcsql17
```

```
Requirement already satisfied: pyodbc in /opt/conda/lib/python3.10/site-packages (5.2.0)
Requirement already satisfied: sqlalchemy in /opt/conda/lib/python3.10/site-packages (2.0.30)
Requirement already satisfied: typing-extensions>=4.6.0 in /opt/conda/lib/python3.10/site-packages (from sqlalchemy) (4.12.2)
Requirement already satisfied: greenlet!=0.4.17 in /opt/conda/lib/python3.10/site-packages (from sqlalchemy) (3.0.3)
Reading package lists... Done
```

Figure 13: Creating Connection Engine

```
from sqlalchemy import create_engine
import sqlalchemy
!pip install pyodbc
!pip install sqlalchemy

# Define connection details
username = 'sqladmin'  # Replace with your Azure SQL admin username
password = 'MINIproject11'        # Replace with your Azure SQL password
server = 'zienzhu2-server.database.windows.net'
database = 'mini-project-db'

# Connection string for Azure SQL Server
connection_string = f"mssql+pyodbc://{username}:{password}@{server}:1433/{database}?driver=ODBC+Driver+17+for+SQL+Server"
engine = create_engine(connection_string)
```

```
Requirement already satisfied: pyodbc in /opt/conda/lib/python3.10/site-packages (5.2.0)
Requirement already satisfied: sqlalchemy in /opt/conda/lib/python3.10/site-packages (2.0.30)
Requirement already satisfied: typing-extensions>=4.6.0 in /opt/conda/lib/python3.10/site-packages (from sqlalchemy) (4.12.2)
Requirement already satisfied: greenlet!=0.4.17 in /opt/conda/lib/python3.10/site-packages (from sqlalchemy) (3.0.3)
```

Figure 14: Connection to Database

- Row/column count screenshots (pg. 7–8)

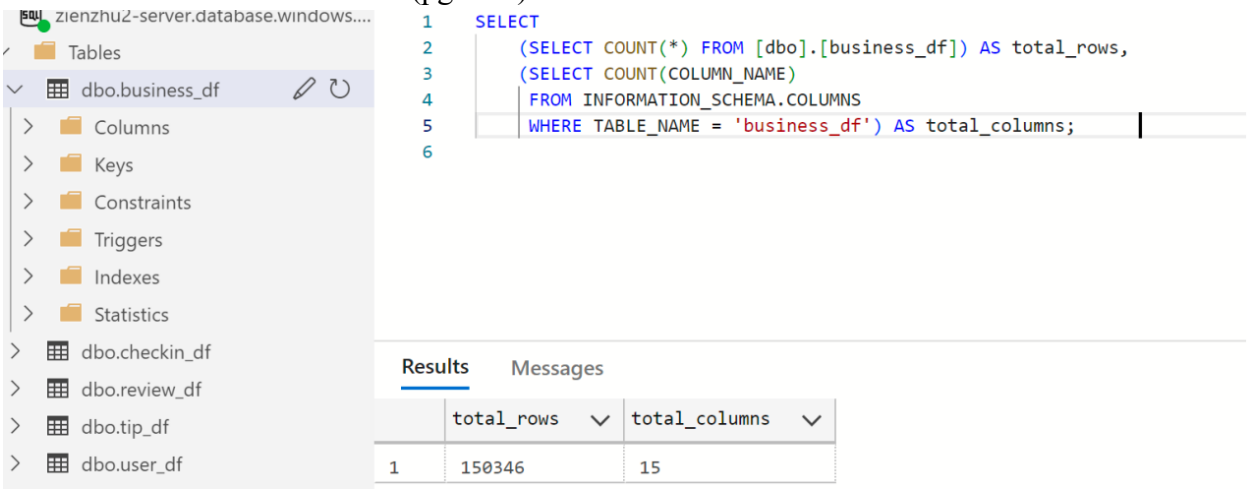


Figure 15: Counts Review – Business

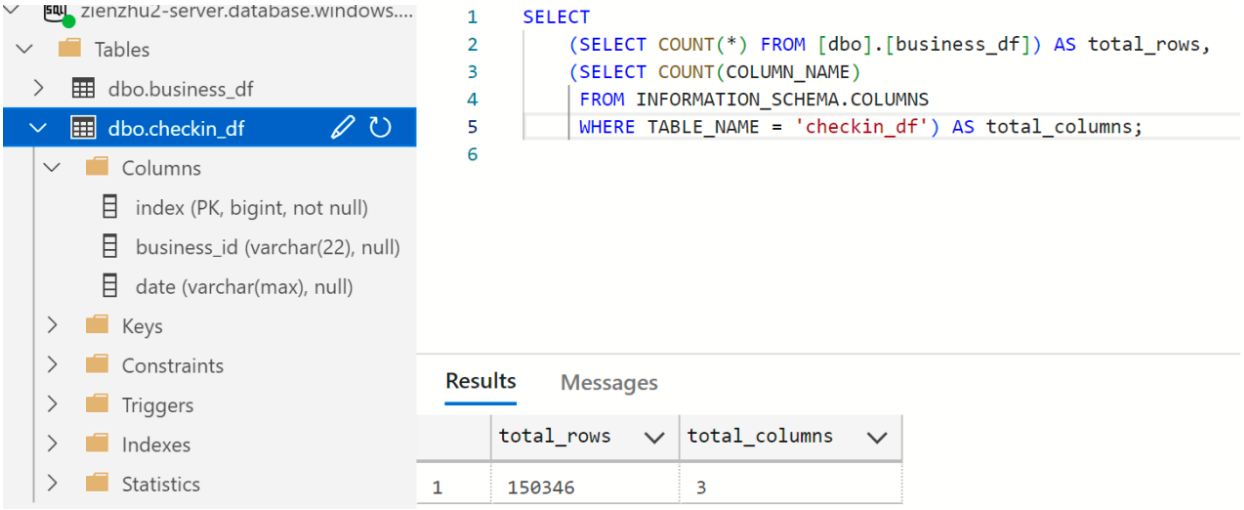


Figure 16: Counts Review – Checkin

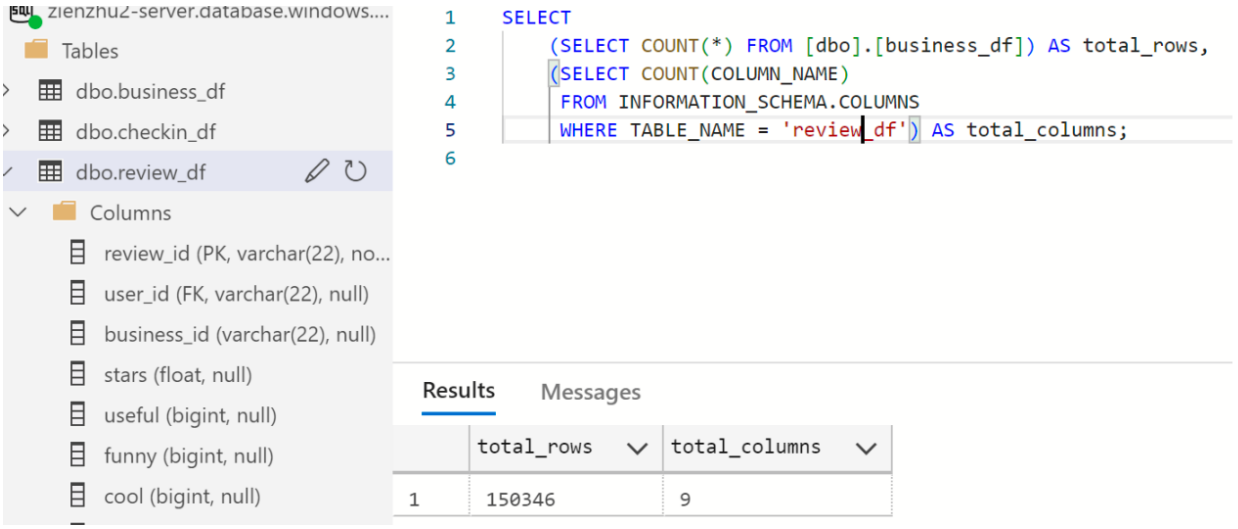


Figure 17: Counts Review – Review

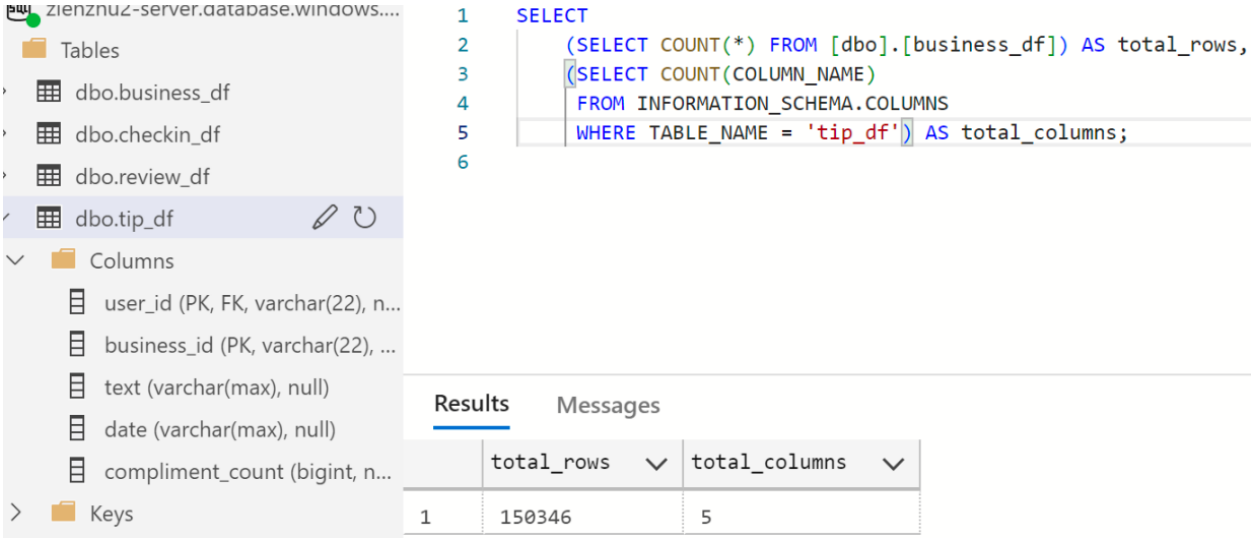


Figure 18: Counts Review – Tip

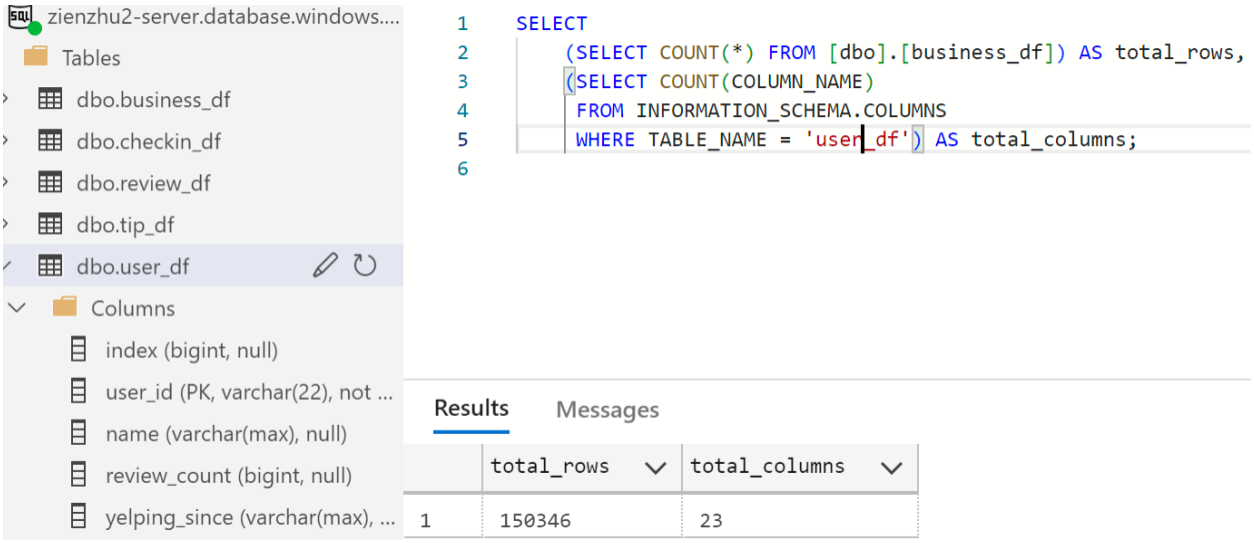


Figure 19: Counts Review - User

- PK/FK setup screenshots

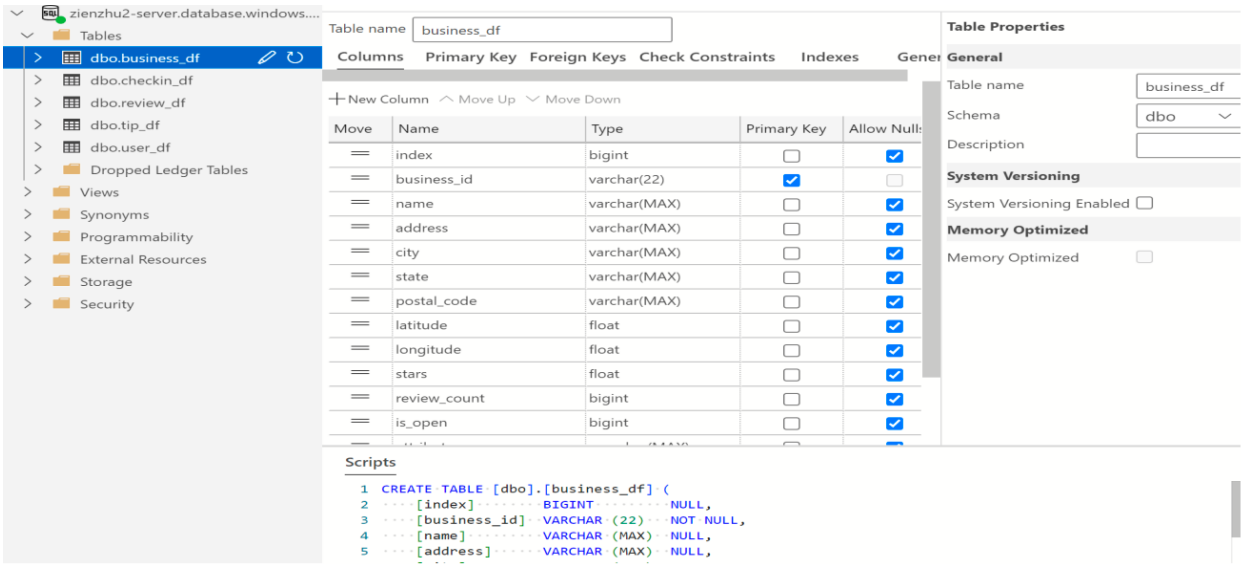


Figure 20: PK/FK Set up – Business

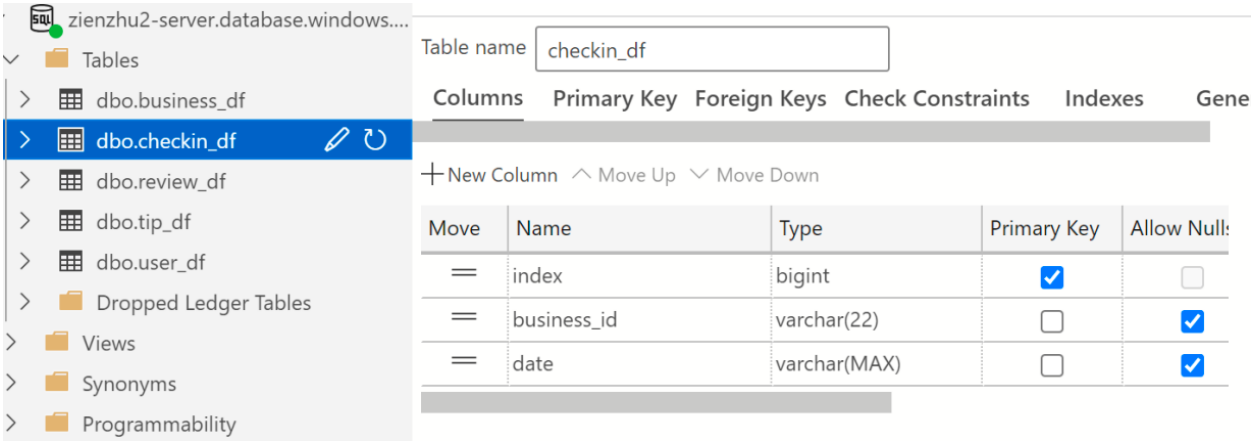


Figure 21: PK/FK Set up - Checkin

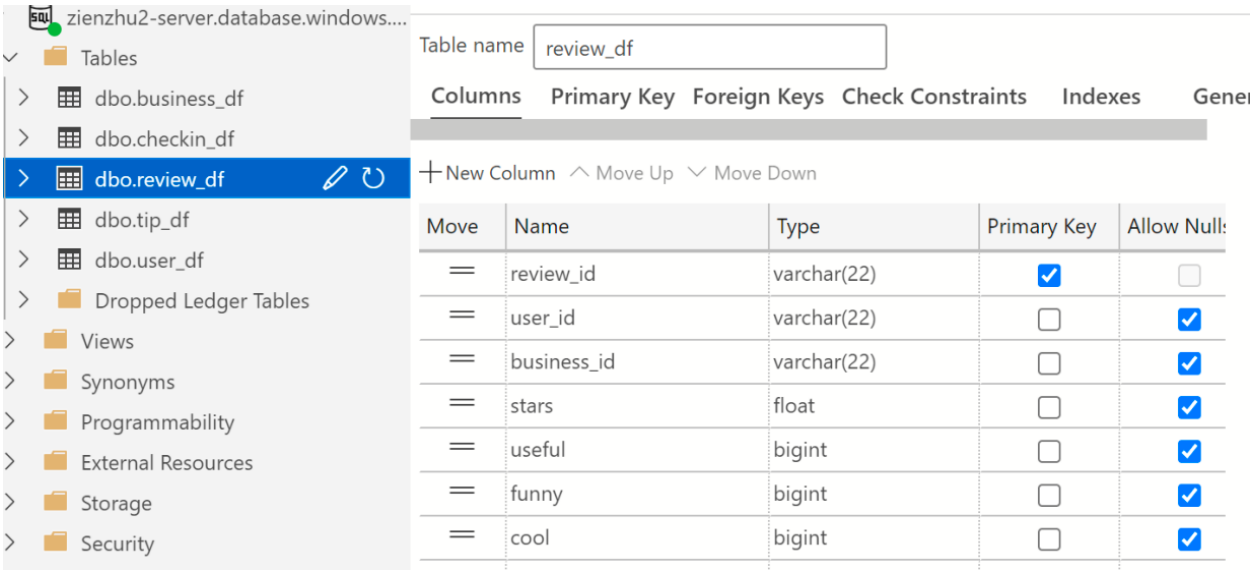


Figure 22: PK/FK Set up – Review

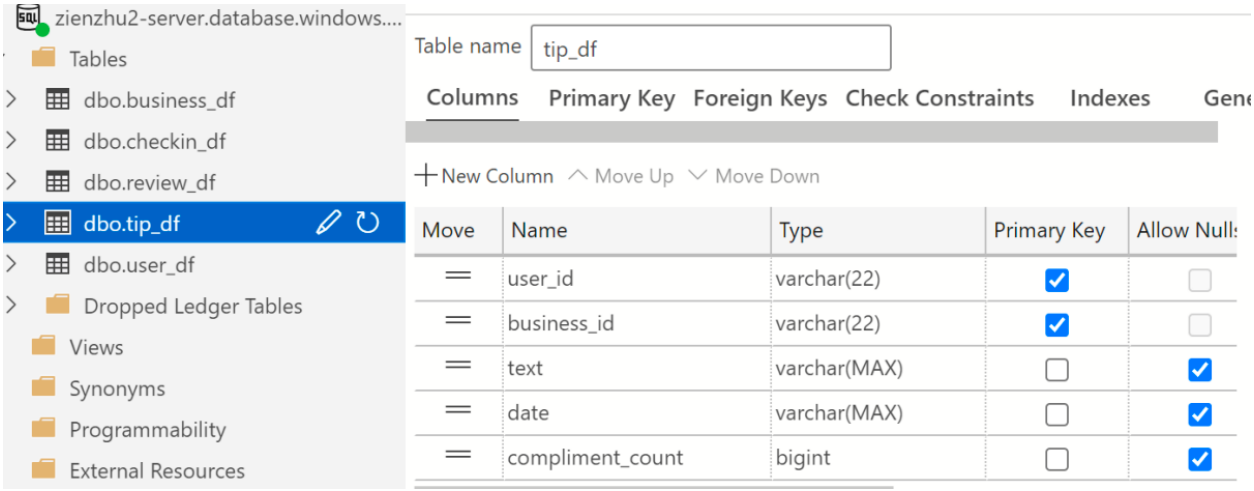


Figure 23: PK/FK Set up – Tip

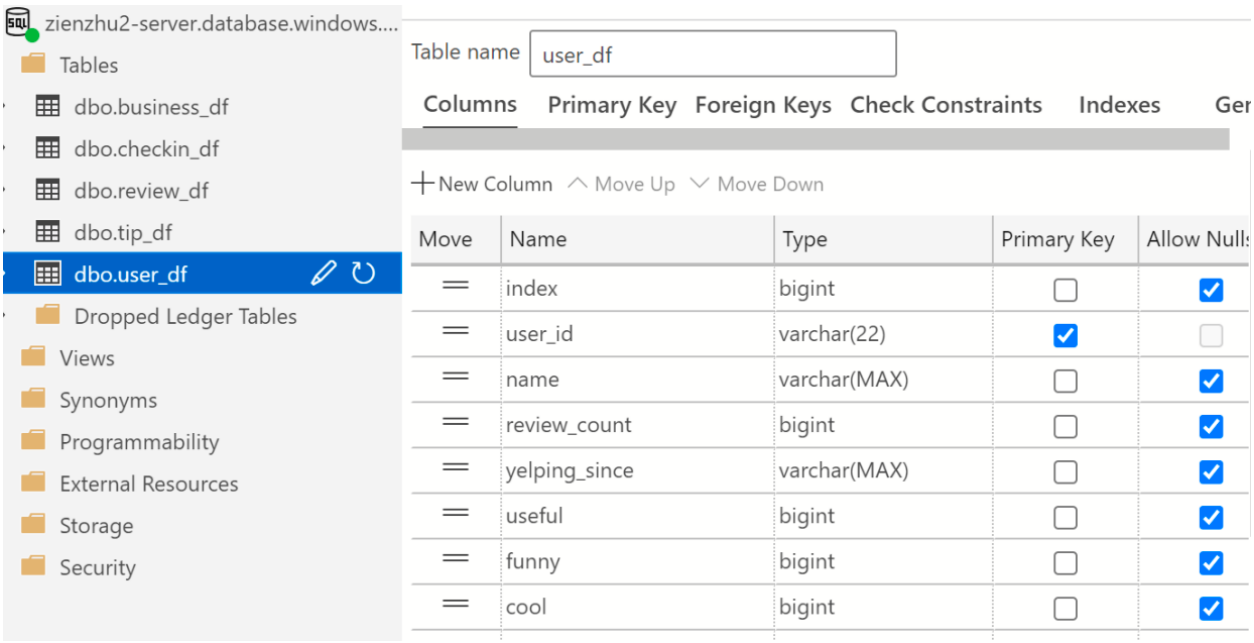


Figure 24: PK/FK Set up - User

- SQL query outputs (pg. 21–24)

```
%%sql
SELECT TOP 20 city, count(DISTINCT business_id) as Restaurant_count
FROM business_df
WHERE UPPER(business_df.categories) LIKE ('%RESTAURANT%')
GROUP BY city
ORDER BY Restaurant_count DESC;
```

```
 * mssql+pyodbc://sqladmin:***@zienzhu2-server.database.windows.net:1433/mini-project-db?driver=ODBC+Driver+17+for+SQL+Server
Done.
```

city	Restaurant_count
Philadelphia	5857
Tampa	2968
Indianapolis	2863
Nashville	2506
Tucson	2473
New Orleans	2262
Edmonton	2168
Saint Louis	1791
Reno	1290
Boise	850

Figure 25: Cities with Most Restaurants

```
%%sql
SELECT TOP 20 city, count(review_df.stars) as reviewer_count,round(AVG(review_df.stars),2) AS average_rating
FROM review_df
JOIN business_df ON review_df.business_id = business_df.business_id
WHERE UPPER(business_df.categories) LIKE ('%RESTAURANT%')
AND city IN ('Philadelphia','Tampa','Indianapolis','Nashville','Tucson','New Orleans','Edmonton','Saint Louis','Reno','Boise',
'Santa Barbara','Clearwater','Wilmington','St. Louis','Metairie','Saint Petersburg','Franklin','St. Petersburg','Sparks','Brandon')
GROUP BY city
ORDER BY average_rating DESC;
```

```
 * mssql+pyodbc://sqladmin:***@zienzhu2-server.database.windows.net:1433/mini-project-db?driver=ODBC+Driver+17+for+SQL+Server
Done.
```

city	reviewer_count	average_rating
Santa Barbara	162283	3.98
New Orleans	476572	3.95
Saint Petersburg	52885	3.93
Clearwater	56443	3.89
St. Petersburg	36179	3.88
Nashville	325787	3.87
Indianapolis	250642	3.86
St. Louis	46010	3.86
Boise	66644	3.85
Tampa	303698	3.84
Saint Louis	178460	3.84
Philadelphia	687571	3.81
Reno	200672	3.78
Tucson	249863	3.74
Brandon	30145	3.7

Figure 26: Average Restaurants Ratings of Cities

```
%%sql
SELECT TOP 20 categories, Round(AVG(review_df.stars),2) AS average_rating
FROM review_df
JOIN business_df ON review_df.business_id = business_df.business_id
WHERE UPPER(business_df.categories) LIKE ('%RESTAURANT%') AND city = 'Tucson'
GROUP BY categories
ORDER BY average_rating DESC;
```

```
 * mssql+pyodbc://sqladmin:***@zienzhu2-server.database.windows.net:1433/mini-project-db?driver=ODBC+Driver+17+for+SQL+Server
Done.
```

categories	average_rating
Brasseries, Comfort Food, Restaurants, Food, Pizza, Sandwiches, Delis	5.0
Mexican, Cafes, Restaurants, Breakfast & Brunch	5.0
Food Trucks, Mexican, Restaurants, Hot Dogs, Food	5.0
Caterers, Restaurants, Cultural Center, Cafes, Arts & Entertainment, Event Planning & Services, Local Services, Venues & Event Spaces	5.0
Event Planning & Services, Caterers, Food, Sandwiches, American (Traditional), Restaurants, Food Trucks, Barbeque, Street Vendors	5.0
Street Vendors, Food Trucks, Food, Mexican, Restaurants	5.0
Burgers, Food Trucks, Restaurants, Food	5.0
Shopping, Food, Wholesale Stores, Restaurants, Specialty Food, Cafes, Coffee Roasteries, Coffee & Tea	5.0
Tattoo, Cafes, Beauty & Spas, Restaurants	5.0
Puerto Rican, Caribbean, Restaurants, Desserts, Food Trucks, Food	5.0
Food, Soup, Desserts, Gluten-Free, Vegan, Bakeries, Sandwiches, Restaurants	5.0
American (Traditional), Food, Food Trucks, Restaurants, Barbeque	5.0
Shaved Ice, Food Stands, Restaurants, Street Vendors, Food Trucks, Food, Desserts	5.0
Trainers, Active Life, Gyms, Health Markets, Restaurants, Food Delivery Services, Specialty Food, Food, Fitness & Instruction, Food Trucks, Event Planning & Services, Juice Bars & Smoothies, Personal Chefs	5.0
Restaurants, Cafes, Juice Bars & Smoothies, Food, Street Vendors, Coffee & Tea	4.97

Figure 27: Top Restaurants Categories by Ratings

```
%%sql
SELECT TOP 10 categories, COUNT(*) AS review_count
FROM review_df
JOIN business_df ON review_df.business_id = business_df.business_id
WHERE UPPER(business_df.categories) LIKE ('%RESTAURANT%') AND city = 'Tucson'
GROUP BY categories
ORDER BY review_count DESC;
```

```
 * mssql+pyodbc://sqladmin:***@zienzhu2-server.database.windows.net:1433/mini-project-db?driver=ODBC+Driver+17+for+SQL+Server
Done.
```

categories	review_count
Restaurants, Mexican	6258
Mexican, Restaurants	5913
Restaurants, Cocktail Bars, Bars, Comfort Food, American (New), Sandwiches, Nightlife, Breakfast & Brunch	2230
Restaurants, American (New)	2078
Tapas/Small Plates, Desserts, Cocktail Bars, Nightlife, Bars, Tacos, Mexican, Food, Restaurants, Salad	1617
Chinese, Restaurants	1349
Restaurants, Chinese	1340
American (Traditional), Restaurants, Mexican, Latin American	1331
Ice Cream & Frozen Yogurt, American (New), Nightlife, Sandwiches, Bars, Restaurants, Salad, American (Traditional), Desserts, Food	1329
Italian, Restaurants	1327

Figure 28: Top Reviewed Categories

```
%%sql

SELECT TOP 20 word, COUNT(*) AS frequency
FROM (
        SELECT value AS word, review_df.business_id
        FROM review_df
        CROSS APPLY STRING_SPLIT(text, ' ')    -- This splits the 'text' column into words
        WHERE stars >= 4
) AS positive_words
JOIN business_df ON positive_words.business_id = business_df.business_id
WHERE UPPER(business_df.categories) LIKE '%RESTAURANT%'
AND city = 'Tucson'
AND UPPER(business_df.categories) LIKE '%MEXICAN%'
AND upper(word) NOT IN ('  ','I','THE','AND','A','TO','IS','WAS','OF','IN','BUT','ON','FOR','IT','W
'YOU','THEY','ARE','HAD','MY','IF','AN','HAVE','WERE','SO','GOOD','GREAT','VERY','NOT','AS','BE','AT','
'OUR','HERE','LIKE','BEST','ONE','JUST','FROM','WILL','GET','THERE','REALLY','OUT','GO','LOVE','ALWAYS'
,'YOUR','WOULD','SHOULD','MIGHT','HATE','POOR','OR','HAS','ABOUT','CAN','BACK','NO','SHE','HE','US','GO
'BY','THEN','SAID','WENT','OTHER','IT_S','%DON''T%','DO NOT','%DID''NT%','ASKED','MORE','ASKED','TOLD'
'-','TRY','LITTLE')
GROUP BY word
ORDER BY frequency DESC;
```

```
 * mssql+pyodbc://sqladmin:***@zienzhu2-server.database.windows.net:1433/mini-project-db?drive
Done.
```

word	frequency
food	20086
place	13818
Mexican	9544
it's	7796
salsa	7760
service	7607
tacos	7133
I've	5256
definitely	5222
restaurant	5093
Tucson	4926
ordered	4867

Figure 29: Words Frequency of High Rating Reviews

```
%%sql

SELECT TOP 20 word, COUNT(*) AS frequency
FROM (
        SELECT value AS word, review_df.business_id
        FROM review_df
        CROSS APPLY STRING_SPLIT(text, ' ')    -- This splits the 'text' column into words
        WHERE stars < 3
) AS negative_words
JOIN business_df ON negative_words.business_id = business_df.business_id
WHERE UPPER(business_df.categories) LIKE '%RESTAURANT%'
AND city = 'Tucson'
AND UPPER(business_df.categories) LIKE '%MEXICAN%'
AND upper(word) NOT IN (' ','I','THE','AND','A','TO','IS','WAS','OF','IN','BUT','ON','FOR','IT','WITH','TH
'YOU','THEY','ARE','HAD','MY','IF','AN','HAVE','WERE','SO','GOOD','GREAT','VERY','NOT','AS','BE','AT','THEIR','
'OUR','HERE','LIKE','BEST','ONE','JUST','FROM','WILL','GET','THERE','REALLY','OUT','GO','LOVE','ALWAYS','ALSO',
,'YOUR','WOULD','SHOULD','MIGHT','HATE','POOR','OR','HAS','ABOUT','CAN','BACK','NO','SHE','HE','US','GOT','EVEN
'BY','THEN','SAID','WENT','OTHER','IT_S','%DON''T%','DO NOT','%DID''NT%','ASKED','MORE','ASKED','TOLD','MENTIO
GROUP BY word
ORDER BY frequency DESC;
```

```
 * mssql+pyodbc://sqladmin:***@zienzhu2-server.database.windows.net:1433/mini-project-db?driver=ODBC+
Done.
```

word	frequency
food	9339
place	4510
ordered	3700
service	3680
order	3121
Mexican	2765
Don't	2642
didn't	2537
restaurant	2354
It's	2176
minutes	2137
salsa	2046
tacos	1810
two	1784
do	1783
taco	1755
going	1746

Figure 30: Words Frequency of Low Rating Reviews

```sql
%%sql
SELECT
        CASE
                WHEN DATEPART(WEEKDAY, CONVERT(DATETIME,
                        SUBSTRING(checkin_df.date, 1, CHARINDEX(',', checkin_df.date + ',') - 1))) = 1 THEN 'Sunday'
                WHEN DATEPART(WEEKDAY, CONVERT(DATETIME,
                        SUBSTRING(checkin_df.date, 1, CHARINDEX(',', checkin_df.date + ',') - 1))) = 2 THEN 'Monday'
                WHEN DATEPART(WEEKDAY, CONVERT(DATETIME,
                        SUBSTRING(checkin_df.date, 1, CHARINDEX(',', checkin_df.date + ',') - 1))) = 3 THEN 'Tuesday'
                WHEN DATEPART(WEEKDAY, CONVERT(DATETIME,
                        SUBSTRING(checkin_df.date, 1, CHARINDEX(',', checkin_df.date + ',') - 1))) = 4 THEN 'Wednesday'
                WHEN DATEPART(WEEKDAY, CONVERT(DATETIME,
                        SUBSTRING(checkin_df.date, 1, CHARINDEX(',', checkin_df.date + ',') - 1))) = 5 THEN 'Thursday'
                WHEN DATEPART(WEEKDAY, CONVERT(DATETIME,
                        SUBSTRING(checkin_df.date, 1, CHARINDEX(',', checkin_df.date + ',') - 1))) = 6 THEN 'Friday'
                WHEN DATEPART(WEEKDAY, CONVERT(DATETIME,
                        SUBSTRING(checkin_df.date, 1, CHARINDEX(',', checkin_df.date + ',') - 1))) = 7 THEN 'Saturday'
        END AS weekday_of_week,
        COUNT(*) AS checkin_count
FROM
        checkin_df
JOIN
        business_df ON checkin_df.business_id = business_df.business_id
WHERE
        business_df.categories LIKE '%RESTAURANT%'
        AND city = 'Tucson'
        AND business_df.categories LIKE '%MEXICAN%'
GROUP BY
        DATEPART(WEEKDAY, CONVERT(DATETIME,
                SUBSTRING(checkin_df.date, 1, CHARINDEX(',', checkin_df.date + ',') - 1)))
ORDER BY
        checkin_count DESC;
```

```
 * mssql+pyodbc://sqladmin:***@zienzhu2-server.database.windows.net:1433/mini-project-db?driver=ODBC+Driver+17+for+SQL+Server
Done.
```

weekday_of_week	checkin_count
Saturday	117
Friday	84
Sunday	76
Tuesday	64
Wednesday	60
Thursday	57
Monday	48

Figure 31: Daily Check-in Counts of the Week

```sql
%%sql
SELECT
        DATEPART(HOUR, CONVERT(DATETIME,
                SUBSTRING(checkin_df.date, 1, CHARINDEX(',', checkin_df.date + ',') - 1))) AS hour_of_day,
        COUNT(*) AS checkin_count
FROM
        checkin_df
JOIN
        business_df ON checkin_df.business_id = business_df.business_id
WHERE
        business_df.categories LIKE '%RESTAURANT%'
        AND city = 'Tucson'
        AND business_df.categories LIKE '%MEXICAN%'
GROUP BY
        DATEPART(HOUR, CONVERT(DATETIME,
                SUBSTRING(checkin_df.date, 1, CHARINDEX(',', checkin_df.date + ',') - 1)))
ORDER BY
        checkin_count DESC;
```

```
 * mssql+pyodbc://sqladmin:***@zienzhu2-server.database.windows.net:1433/mini-project-db?driver=ODBC+Driver+17+for+SQL+Server
Done.
```

hour_of_day	checkin_count
19	58
1	58
2	57
18	50
20	36
23	32
3	31
0	30
22	28
17	23
21	22
4	17
5	13
6	11
15	10
16	9
7	6
14	5
9	4
12	2
8	2
13	1
10	1

Figure 32: Hourly Check-in Counts

```
[ ] %%sql
    SELECT TOP 10 user_id, COUNT(*) AS total_reviews, AVG(review_df.stars) AS avg_user_rating
    FROM review_df
    JOIN business_df ON review_df.business_id = business_df.business_id
    WHERE UPPER(business_df.categories) LIKE ('%RESTAURANT%')
    AND UPPER(business_df.categories) LIKE ('%MEXICAN%')
    AND city = 'Tucson'
    GROUP BY user_id
    ORDER BY total_reviews DESC, avg_user_rating DESC;
```

```
 * mssql+pyodbc://sqladmin:***@zienzhu2-server.database.windows.net:1433/mini-project-db?driver=ODBC+Driver+17+for+SQL+Server
Done.
```

user_id	total_reviews	avg_user_rating
EAHCCgMFoJxlOPcCPQ91SQ	81	3.802469135802469
BrH27yHhgDEylDZfFdJvSQ	78	2.717948717948718
-rGU3wo4fgEnGRboYdMTjw	67	3.582089552238806
2YKkLFeOx-0zRcWp0KUv_Q	67	3.3582089552238807
PAc93PtEbYDtytBQ9Dyjug	66	3.757575757575758
dvu83QWKsnnbGvNVQ3mv7A	66	3.742424242424242
6ObFF8-uKnOAlXuSH4TlyQ	65	4.1692307692307695
xWmYN57XXZbg0LOK8WbbFQ	63	4.825396825396825
IeMOR7VSm5z-0r60iJ90EA	61	3.8360655737704916
9Y1YkloHk2MAE3hkwYdFKA	58	3.603448275862069

Figure 33: Most Engaged Users

```
[ ] %%sql
    SELECT business_df.city,
    Round(AVG(user_df.average_stars),2) AS avg_user_rating, SUM(tip_df.compliment_count) AS total_compliments, COUNT(DISTINCT tip_df.business_id) AS businesses_tippe
    FROM user_df
    JOIN tip_df ON user_df.user_id = tip_df.user_id
    JOIN business_df ON tip_df.business_id = business_df.business_id
    WHERE UPPER(business_df.categories) LIKE '%RESTAURANT%' AND UPPER(business_df.categories) LIKE '%MEXICAN%'
    AND business_df.city = 'Tucson'
    GROUP BY business_df.city
    ORDER BY avg_user_rating DESC;
```

```
 * mssql+pyodbc://sqladmin:***@zienzhu2-server.database.windows.net:1433/mini-project-db?driver=ODBC+Driver+17+for+SQL+Server
Done.
```

city	avg_user_rating	total_compliments	businesses_tipped
Tucson	3.8	118	471

Figure 34: User Tip Engagement