BDE Solution

Bde Candidate

Introduction

- The given datasets are:
    1. website_dataset.csv
    2. facebook_dataset.csv
    3. google_dataset.csv
- The main task:
    1. combine data from 3 given input datasets
    2. in a most accurate way,
    3. cleaned and transformed data.

Terminology & Details

- **Language and environment:** Pyspark (python+spark) - jupyter notebook – databricks
- *web_data, website_data* - referring to the website dataset (website_dataset.csv)
- *fb_data, facebook_data* - referring to the facebook dataset (facebook_dataset.csv)
- *gg_data, google_data* - referring to the google dataset (google_dataset.csv)
- *{dataset}_missing* - referring to null value percentage of dataset for each column
- *{dataset}_distinct* - referring to unique value percentage of dataset for each column
- *web_fb* - website_dataset and facebook_dataset
- *web_fb_left* - website_dataset and facebook_dataset join with left operation
- *web_fb_inner* - website_dataset and facebook_dataset join with inner operation
- *remaining_gg* - non-matched data in google_dataset after join operation
- *web_fb_gg_full* - website_dataset, facebook_dataset, google_dataset are merged into that dataset

Data Reading

- Problem (new line characters):
  1. multiline option
  2. encoding option with UTF-8

Python

```python
website_data = spark.read.option("delimiter", ";").option("encoding","UTF-8").option("multiline",True).csv(website_dataset_filepath,
header='true', inferSchema='true')
website_data.count()
```

(4) Spark Jobs

website_data: pyspark.sql.dataframe.DataFrame = [root_domain: string, domain_suffix: string ... 9 more fields]

```
72018
```

Understanding Data

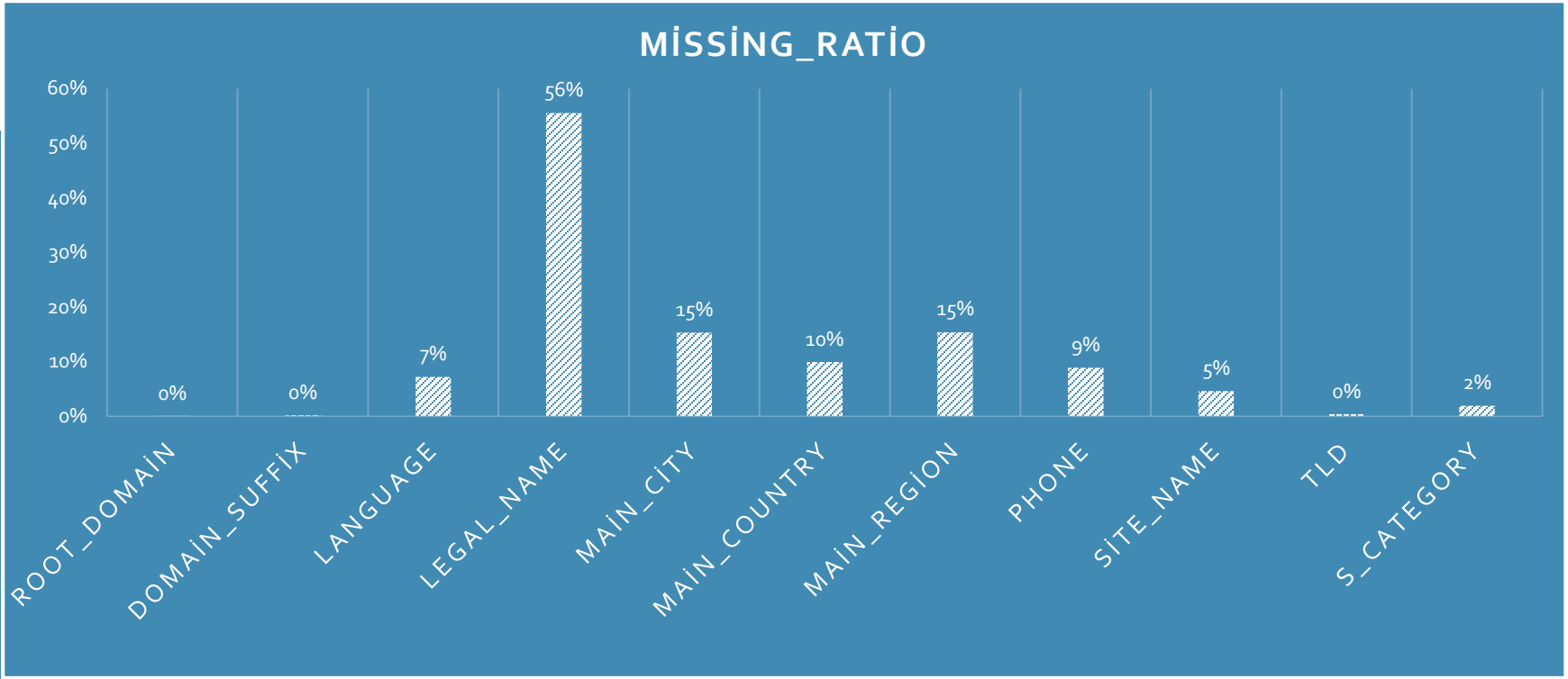
```
1  website_data.printSchema()

root
 |-- root_domain: string (nullable = true)
 |-- domain_suffix: string (nullable = true)
 |-- language: string (nullable = true)
 |-- legal_name: string (nullable = true)
 |-- main_city: string (nullable = true)
 |-- main_country: string (nullable = true)
 |-- main_region: string (nullable = true)
 |-- phone: string (nullable = true)
 |-- site_name: string (nullable = true)
 |-- tld: string (nullable = true)
 |-- s_category: string (nullable = true)
```

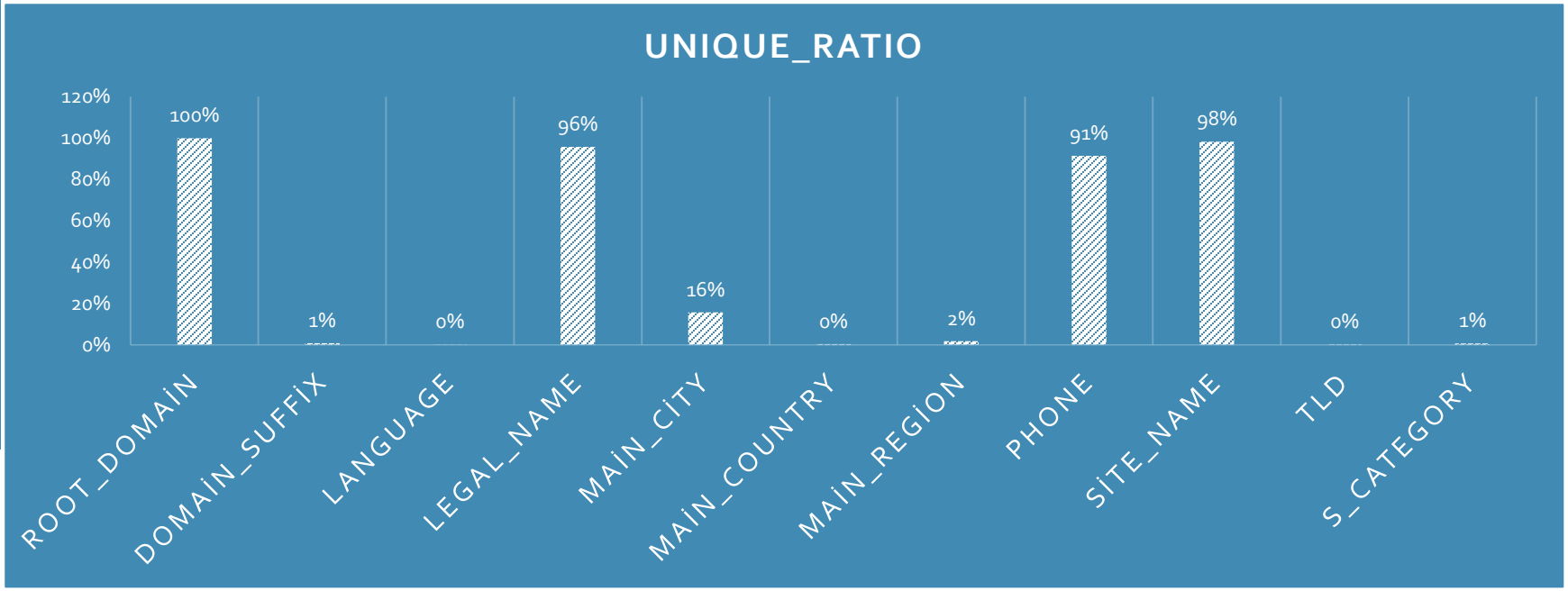
```
1  facebook_data.printSchema()

root
 |-- domain: string (nullable = true)
 |-- address: string (nullable = true)
 |-- categories: string (nullable = true)
 |-- city: string (nullable = true)
 |-- country_code: string (nullable = true)
 |-- country_name: string (nullable = true)
 |-- description: string (nullable = true)
 |-- email: string (nullable = true)
 |-- link: string (nullable = true)
 |-- name: string (nullable = true)
 |-- page_type: string (nullable = true)
 |-- phone: long (nullable = true)
 |-- phone_country_code: string (nullable = true)
 |-- region_code: string (nullable = true)
 |-- region_name: string (nullable = true)
 |-- zip_code: string (nullable = true)
```

```
1  google_data.printSchema()

root
 |-- address: string (nullable = true)
 |-- category: string (nullable = true)
 |-- city: string (nullable = true)
 |-- country_code: string (nullable = true)
 |-- country_name: string (nullable = true)
 |-- name: string (nullable = true)
 |-- phone: long (nullable = true)
 |-- phone_country_code: string (nullable = true)
 |-- raw_address: string (nullable = true)
 |-- raw_phone: string (nullable = true)
 |-- region_code: string (nullable = true)
 |-- region_name: string (nullable = true)
 |-- text: string (nullable = true)
 |-- zip_code: string (nullable = true)
 |-- domain: string (nullable = true)
```

Understanding Data

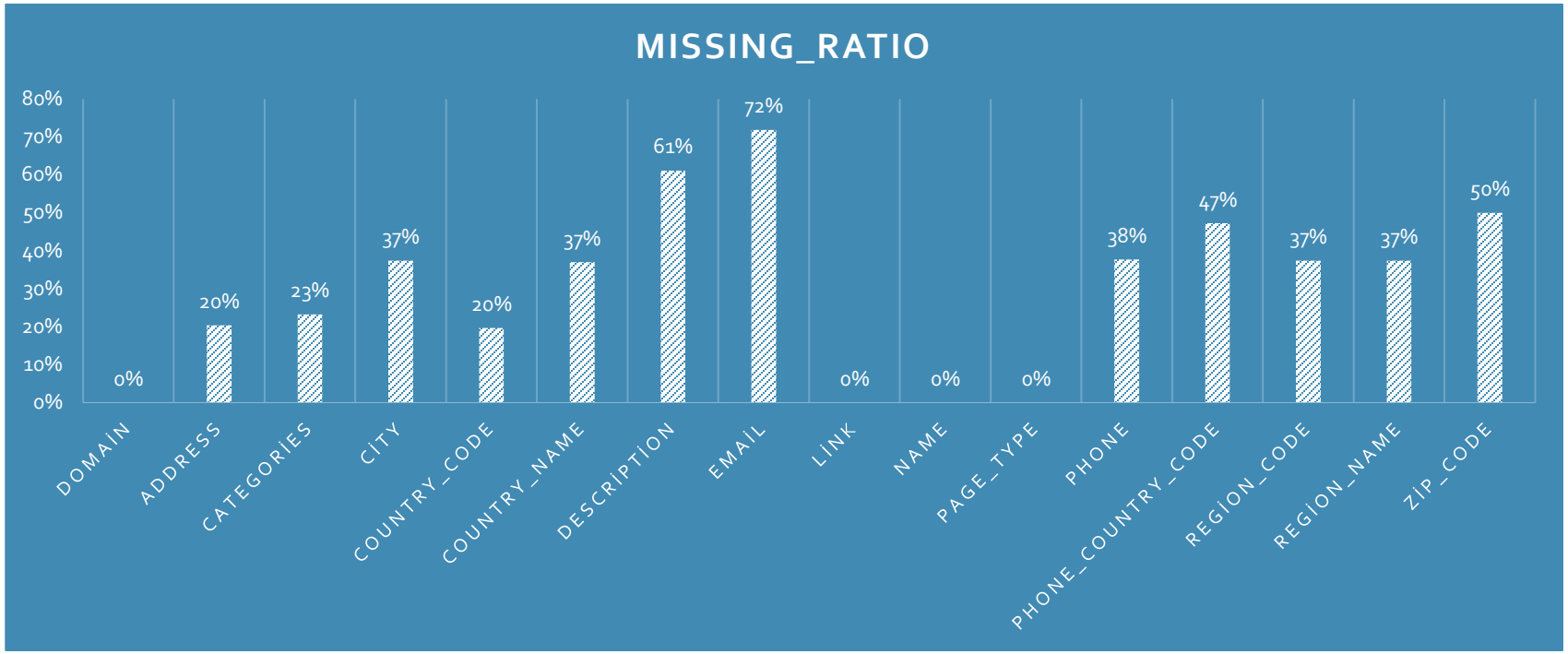
Website_data

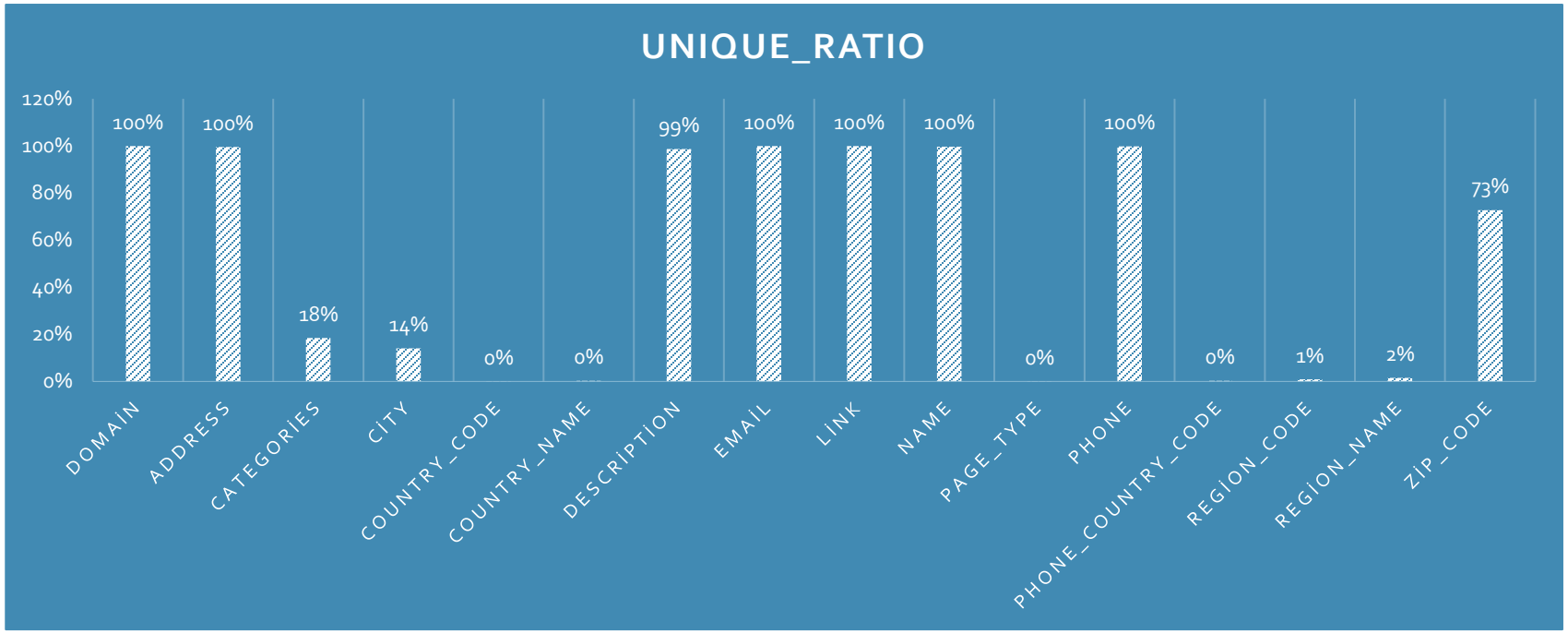




Understanding Data

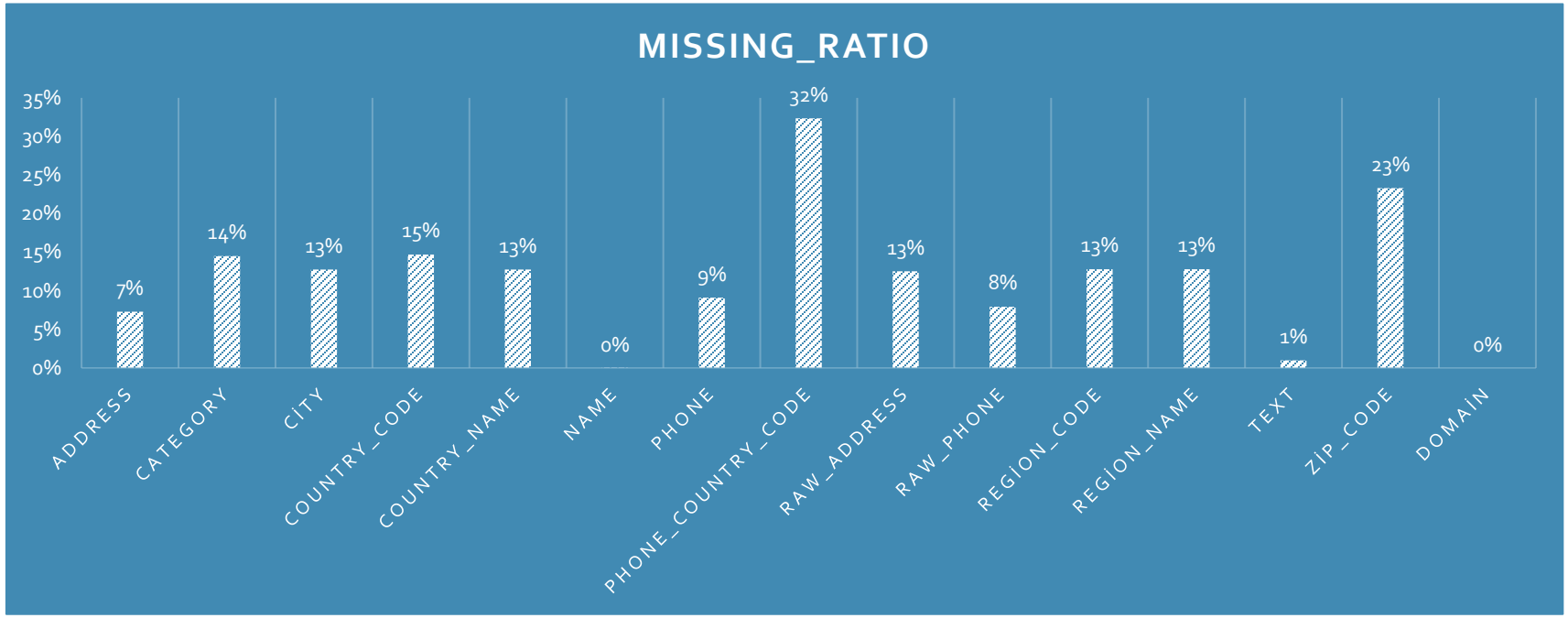
Facebook_data

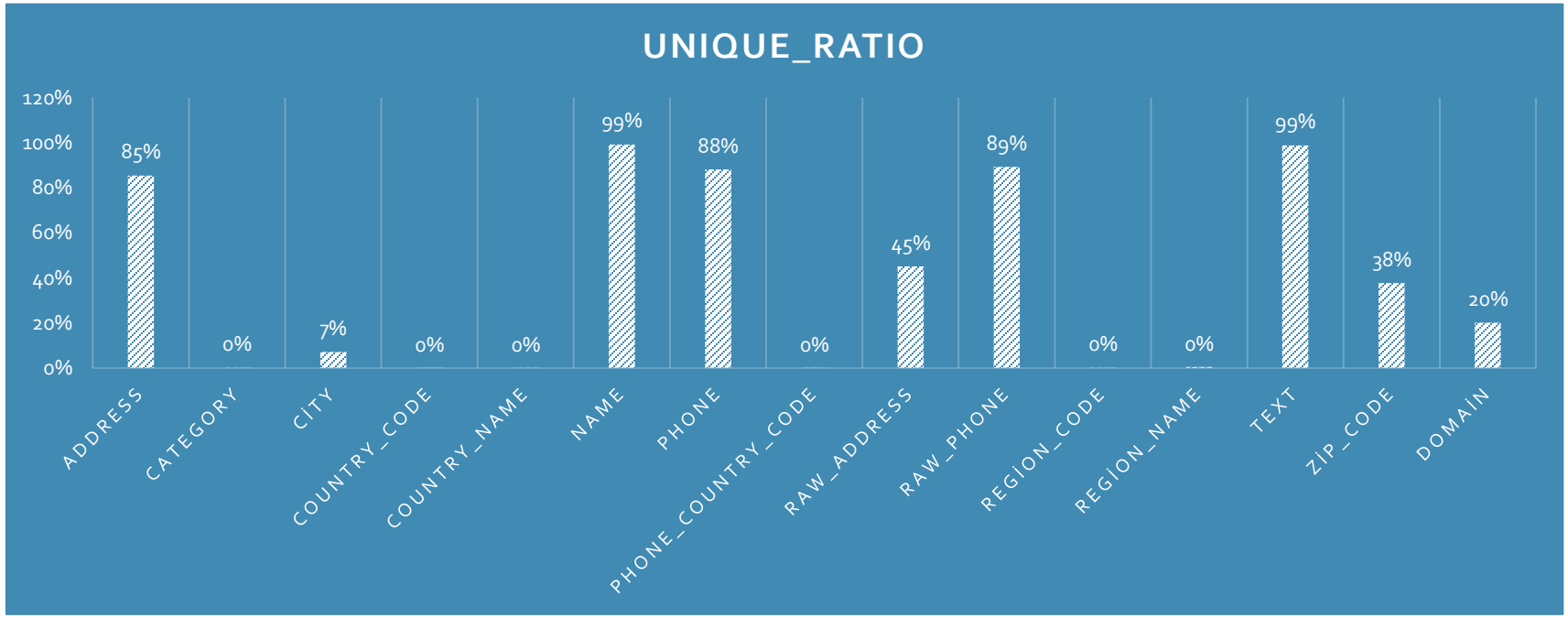




Understanding Data

Google_data





Data Preparation and Merging

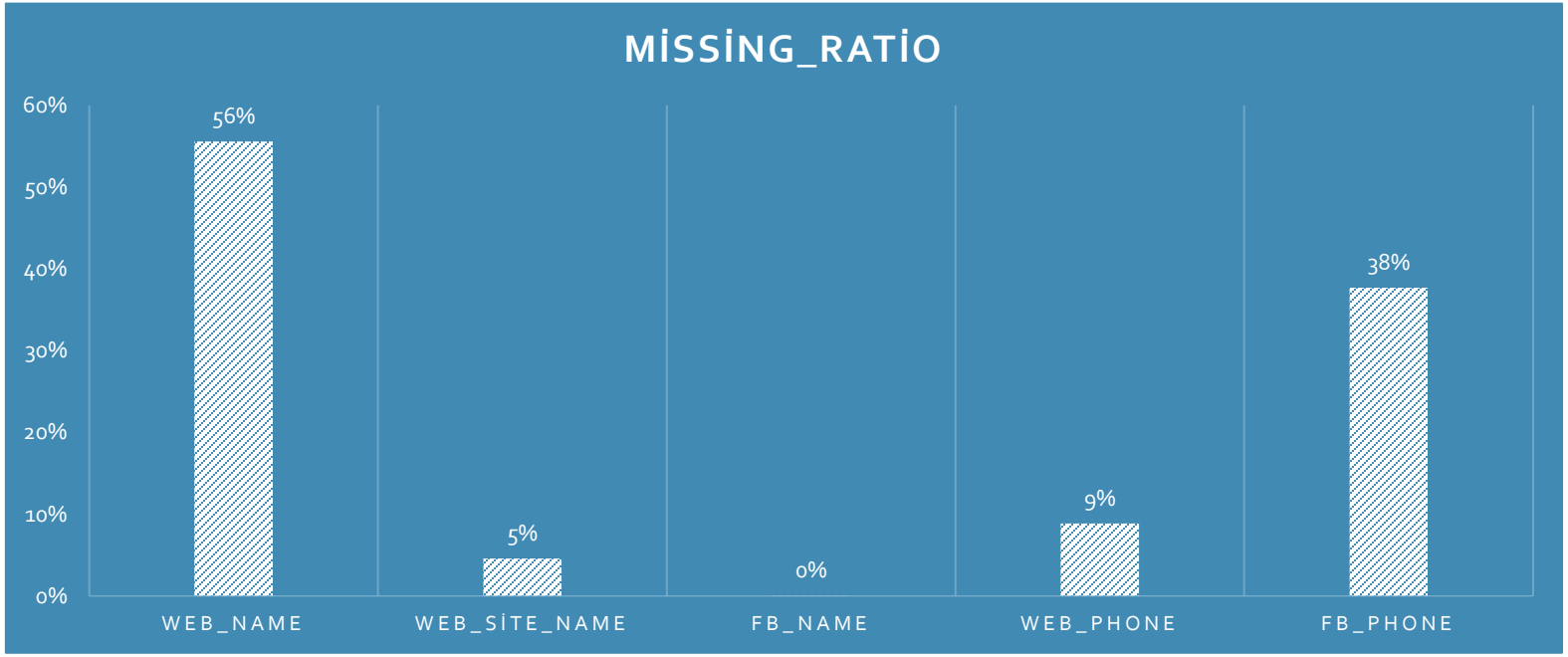
```python
web_fb_inner = web_data.join(fb_data, on = [web_data.web_domain == fb_data.fb_domain], how = "inner")
web_fb_left = web_data.join(fb_data, on = [web_data.web_domain == fb_data.fb_domain], how = "left")
print("join ratio:",web_fb_inner.count()/web_fb_left.count())
```

▸ (6) Spark Jobs

▸ web_fb_inner: pyspark.sql.dataframe.DataFrame = [web_domain: string, web_name: string ... 17 more fields]

▸ web_fb_left: pyspark.sql.dataframe.DataFrame = [web_domain: string, web_name: string ... 17 more fields]

```
join ratio: 0.9998194895720515
```



Data Preparation and Merging

```python
from pyspark.sql import functions as F
# FB phone has well format, if it is available use it. I try both version and fb_phone has bigger coverage rate
web_fb_data = web_fb_data.withColumn('fb_phone', F.regexp_replace(F.col("fb_phone"), '[^0-9]+', ''))
web_fb_data = web_fb_data.withColumn('web_phone', F.regexp_replace(F.col("web_phone"), '[^0-9]+', ''))
web_fb_data = web_fb_data.withColumn('new_phone', F.when(F.isnan("fb_phone") | F.col("fb_phone").isNull(), F.col("web_phone")).otherwise(F.
col("fb_phone")))

gg_data = gg_data.withColumn('gg_phone', F.regexp_replace(F.col("gg_phone"), '[^0-9]+', ''))
```

```python
web_fb_gg_inner = web_fb_data.join(gg_data, on = [web_fb_data.new_phone == gg_data.gg_phone], how = "inner")
web_fb_gg_left = web_fb_data.join(gg_data, on = [web_fb_data.new_phone == gg_data.gg_phone], how = "left")
print("join ratio:",web_fb_gg_inner.count()/web_fb_gg_left.count())
print("domain coverage ratio:",web_fb_gg_inner.select("web_domain").distinct().count()/web_fb_data.select("web_domain").distinct().count())
```

▸ (19) Spark Jobs

▸ web_fb_gg_inner: pyspark.sql.dataframe.DataFrame = [web_domain: string, web_name: string ... 27 more fields]

▸ web_fb_gg_left: pyspark.sql.dataframe.DataFrame = [web_domain: string, web_name: string ... 27 more fields]

```
join ratio: 0.7339806069447763
domain coverage ratio: 0.7062956483101447
```

MinHashLSH Model & approxSimilarityJoin

```python
remaining_web_fb = remaining_web_fb.withColumn('new_name',
  F.when((F.isnan("fb_name") | F.col("fb_name").isNull()) & (F.isnan("web_name") | F.col("web_name").isNull()), F.col("web_site_name"))
  .when(F.isnan("fb_name") | F.col("fb_name").isNull(), F.col("web_name"))
  .otherwise(F.col("fb_name"))
)
```

```python
pipeline = Pipeline(stages=[
        Tokenizer(inputCol="company_name", outputCol="tokens"),
        NGram(n=2, inputCol="tokens", outputCol="ngrams"),
        HashingTF(inputCol="ngrams", outputCol="vectors")
    ])

lsh = MinHashLSH(inputCol="vectors", outputCol="lsh")
```

```python
matched_df = model.approxSimilarityJoin(df, df2, 0.5, "confidence")
print(matched_df.count())
display(matched_df)
```

```python
print("domain coverage ratio for remaining:",remaining_web_fb_gg.select("web_domain").distinct().count()/remaining_web_fb.select("web_domain").distinct().count())
```

▸ (17) Spark Jobs

```
domain coverage ratio for remaining: 0.4181164901664145
```

```python
print("final coverage ratio:",web_fb_gg_full.select("web_domain").distinct().count()/web_fb_data.select("web_domain").distinct().count())
```

▸ (17) Spark Jobs

```
final coverage ratio: 0.8290982809853092
```

Conclusion

- The project successfully merged data from three distinct sources
  - website, Facebook, and Google
- Achieving an 82.9% coverage,
  - this process can be further improved
    - enhancement and optimization
- When I encounter very similar data across sources,
  - I can prioritize data from the source
    - with the highest reliability
    - based on my domain-specific knowledge
    - or previous experiences.

Thank you

BDE candidate