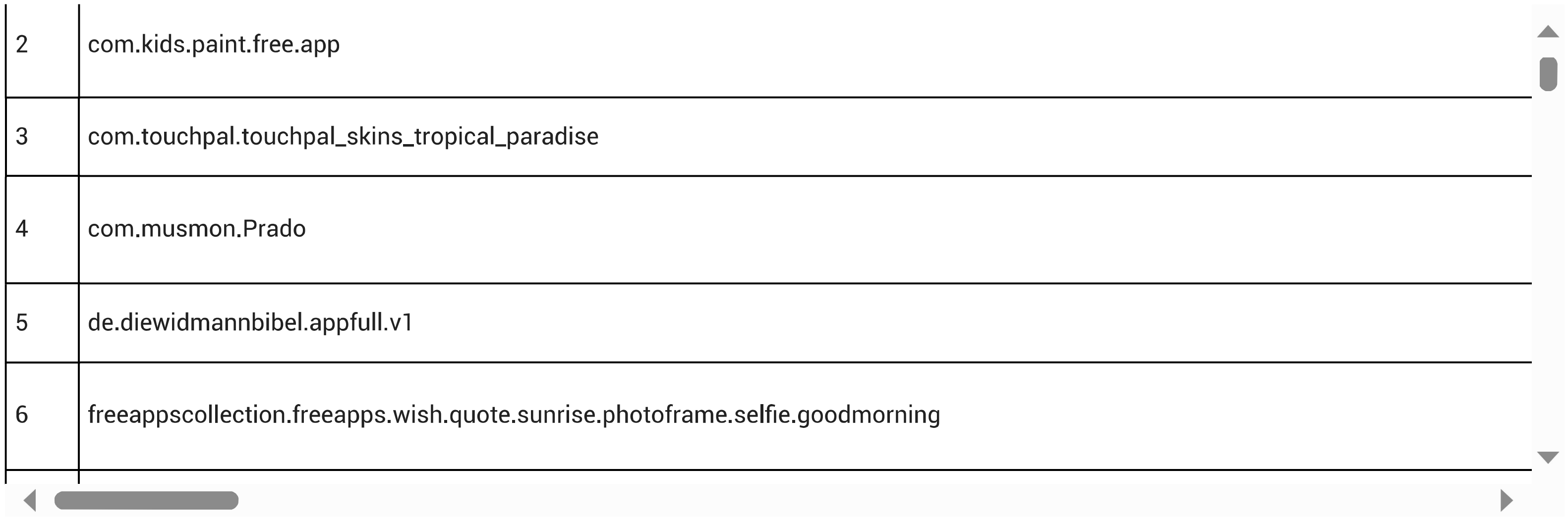
```
df = spark.read.format('csv').option("header",True).option("sep",",").load(r"dbfs:/FileStore/google_play_dataset_by_tapivedotcom.csv")
display(df)
# df.show(3)
```



2	com.kids.paint.free.app
3	com.touchpal.touchpal_skins_tropical_paradise
4	com.musmon.Prado
5	de.diewidmannbibel.appfull.v1
6	freeappscollection.freeapps.wish.quote.sunrise.photoframe.selfie.goodmorning

```
df.printSchema()
```

```
root
 |-- _c0: string (nullable = true)
 |-- appId: string (nullable = true)
 |-- developer: string (nullable = true)
 |-- developerId: string (nullable = true)
 |-- developerWebsite: string (nullable = true)
 |-- free: string (nullable = true)
 |-- genre: string (nullable = true)
 |-- genreId: string (nullable = true)
 |-- inAppProductPrice: string (nullable = true)
 |-- minInstalls: string (nullable = true)
 |-- offersIAP: string (nullable = true)
 |-- originalPrice: string (nullable = true)
 |-- price: string (nullable = true)
 |-- ratings: string (nullable = true)
 |-- len screenshots: string (nullable = true)
 |-- adSupported: string (nullable = true)
 |-- containsAds: string (nullable = true)
 |-- reviews: string (nullable = true)
 |-- releasedDayYear: string (nullable = true)
 |-- sale: string (nullable = true)
 |-- score: string (nullable = true)
 |-- summary: string (nullable = true)
 |-- title: string (nullable = true)
 |-- updated: string (nullable = true)
 |-- histogram1: string (nullable = true)
 |-- histogram2: string (nullable = true)
 |-- histogram3: string (nullable = true)
 |-- histogram4: string (nullable = true)
 |-- histogram5: string (nullable = true)
 |-- releasedDay: string (nullable = true)
 |-- releasedYear: string (nullable = true)
 |-- releasedMonth: string (nullable = true)
 |-- dateUpdated: string (nullable = true)
 |-- minprice: string (nullable = true)
 |-- maxprice: string (nullable = true)
 |-- ParseReleasedDayYear: string (nullable = true)
```

```
from pyspark.sql.types import *
df = df.withColumn("releasedYear",col("releasedYear").cast(IntegerType())).withColumn("price",col("price").cast(IntegerType()))


num_unique_values = df.select("releasedYear").distinct().count()


from pyspark.sql.functions import *
from pyspark.ml.feature import Bucketizer


if num_unique_values >= 20:
    # 5-year ranges
    bin_edges = [2000, 2005, 2010, 2015, 2020]

    bucketizer = Bucketizer(splits=bin_edges, inputCol="releasedYear", outputCol="year_bins")

    df_bucketed = bucketizer.transform(df)

    df_bucketed.select("releasedYear", "year_bins").show()
else:
    print("Not enough unique values for binning.")
```

```
+------------+---------+
|releasedYear|year_bins|
+------------+---------+
|        2014|      2.0|
|        2014|      2.0|
|        2013|      2.0|
|        2014|      2.0|
|        2013|      2.0|
|        2015|      3.0|
|        2016|      3.0|
|        2016|      3.0|
|        2016|      3.0|
|        2016|      3.0|
|        2016|      3.0|
|        2016|      3.0|
|        2016|      3.0|
|        2016|      3.0|
|        2016|      3.0|
|        2017|      3.0|
|        2017|      3.0|
|        2016|      3.0|
|        2017|      3.0|
|        2017|      3.0|
+------------+---------+
only showing top 20 rows
```

```
df_bucketed.printSchema()
```

```
root
 |-- _c0: string (nullable = true)
 |-- appId: string (nullable = true)
 |-- developer: string (nullable = true)
 |-- developerId: string (nullable = true)
 |-- developerWebsite: string (nullable = true)
 |-- free: string (nullable = true)
 |-- genre: string (nullable = true)
 |-- genreId: string (nullable = true)
 |-- inAppProductPrice: string (nullable = true)
 |-- minInstalls: string (nullable = true)
 |-- offersIAP: string (nullable = true)
 |-- originalPrice: string (nullable = true)
 |-- price: integer (nullable = true)
 |-- ratings: string (nullable = true)
 |-- len screenshots: string (nullable = true)
 |-- adSupported: string (nullable = true)
 |-- containsAds: string (nullable = true)
 |-- reviews: string (nullable = true)
 |-- releasedDayYear: string (nullable = true)
 |-- sale: string (nullable = true)
 |-- score: string (nullable = true)
 |-- summary: string (nullable = true)
 |-- title: string (nullable = true)
 |-- updated: string (nullable = true)
 |-- histogram1: string (nullable = true)
 |-- histogram2: string (nullable = true)
 |-- histogram3: string (nullable = true)
 |-- histogram4: string (nullable = true)
 |-- histogram5: string (nullable = true)
 |-- releasedDay: string (nullable = true)
 |-- releasedYear: integer (nullable = true)
 |-- releasedMonth: string (nullable = true)
 |-- dateUpdated: string (nullable = true)
 |-- minprice: string (nullable = true)
 |-- maxprice: integer (nullable = true)
 |-- ParseReleasedDayYear: string (nullable = true)
 |-- year_bins: double (nullable = true)
```

```
from pyspark.sql.functions import count,concat_ws
counts_df = df_bucketed.groupBy("Price", "genre").agg(count("*").alias("count"))


total_count = counts_df.agg(sum("count").alias("total_count")).collect()[0]["total_count"]


filtered_counts_df = counts_df.filter((col("count") / total_count) >= 0.02)


filtered_counts_df.select('count').show(2)
```

```
+------+
| count|
+------+
|230282|
|123949|
+------+
only showing top 2 rows
```

```
filtered_counts_df = filtered_counts_df.withColumn(
    "output",
    concat_ws(
        ";",
        *[concat(col(name), "=", col(name)) for name in filtered_counts_df.columns[:-1]],
        col("count")
    )
)


filtered_counts_df = filtered_counts_df.select(concat_ws(";",*[concat(col(name),"=",col(name)) for name in filtered_counts_df.columns]),col("coun
---------------------------------------------------------------------------
AnalysisException                         Traceback (most recent call last)
File <command-1945143259595152>:1
----> 1 filtered_counts_df = filtered_counts_df.select(concat_ws(";",*[concat(col(name),"=",col(name)) for name in
filtered_counts_df.columns]),col("count"))

File /databricks/spark/python/pyspark/instrumentation_utils.py:48, in _wrap_function.<locals>.wrapper(*args, **kwargs)
     46 start = time.perf_counter()
     47 try:
---> 48     res = func(*args, **kwargs)
     49     logger.log_success(
     50         module_name, class_name, function_name, time.perf_counter() - start, signature
     51     )
     52     return res

File /databricks/spark/python/pyspark/sql/dataframe.py:3023, in DataFrame.select(self, *cols)
   2978 def select(self, *cols: "ColumnOrName") -> "DataFrame":  # type: ignore[misc]
   2979     """Projects a set of expressions and returns a new :class:`DataFrame`.
   2980 
   2981     .. versionadded:: 1.3.0
   (...)
   3021     +-----+---+
   3022     """
-> 3023     jdf = self._jdf.select(self._jcols(*cols))
   3024     return DataFrame(jdf, self.sparkSession)

File /databricks/spark/python/lib/py4j-0.10.9.5-src.zip/py4j/java_gateway.py:1321, in JavaMember.__call__(self, *args)
   1315 command = proto.CALL_COMMAND_NAME +\
   1316     self.command_header +\
   1317     args_command +\
   1318     proto.END_COMMAND_PART
   1320 answer = self.gateway_client.send_command(command)
-> 1321 return_value = get_return_value(
   1322     answer, self.gateway_client, self.target_id, self.name)
   1324 for temp_arg in temp_args:
   1325     temp_arg._detach()

File /databricks/spark/python/pyspark/errors/exceptions.py:234, in capture_sql_exception.<locals>.deco(*a, **kw)
    230 converted = convert_exception(e.java_exception)
    231 if not isinstance(converted, UnknownException):
    232     # Hide where the exception came from that shows a non-Pythonic
    233     # JVM exception message.
--> 234     raise converted from None
    235 else:
    236     raise

AnalysisException: [UNRESOLVED_COLUMN.WITH_SUGGESTION] A column or function parameter with name `=` cannot be
resolved. Did you mean one of the following? [`Price`, `count`, `genre`].;
'Project [unresolvedalias(concat_ws(;, concat(Price#3632, '=, Price#3632), concat(genre#3016, '=, genre#3016),
concat(count#5236L, '=, count#5236L)), Some(org.apache.spark.sql.Column$$Lambda$9179/5191456@164d4db9)), count#5236L]
+- Filter ((cast(count#5236L as double) / cast(3460966 as double)) >= 0.02)
   +- Aggregate [Price#3632, genre#3016], [Price#3632, genre#3016, count(1) AS count#5236L]
      +- Project [_c0#3010, appId#3011, developer#3012, developerId#3013, developerWebsite#3014, free#3015,
genre#3016, genreId#3017, inAppProductPrice#3018, minInstalls#3019, offersIAP#3020, originalPrice#3021, price#3632,
ratings#3023, len screenshots#3024, adSupported#3025, containsAds#3026, reviews#3027, releasedDayYear#3028, sale#3029,
score#3030, summary#3031, title#3032, updated#3033, ... 13 more fields]
         +- Project [_c0#3010, appId#3011, developer#3012, developerId#3013, developerWebsite#3014, free#3015,
genre#3016, genreId#3017, inAppProductPrice#3018, minInstalls#3019, offersIAP#3020, originalPrice#3021,
cast(price#3022 as int) AS price#3632, ratings#3023, len screenshots#3024, adSupported#3025, containsAds#3026,
reviews#3027, releasedDayYear#3028, sale#3029, score#3030, summary#3031, title#3032, updated#3033, ... 12 more fields]
            +- Project [_c0#3010, appId#3011, developer#3012, developerId#3013, developerWebsite#3014, free#3015,
genre#3016, genreId#3017, inAppProductPrice#3018, minInstalls#3019, offersIAP#3020, originalPrice#3021, price#3022,
ratings#3023, len screenshots#3024, adSupported#3025, containsAds#3026, reviews#3027, releasedDayYear#3028, sale#3029,
score#3030, summary#3031, title#3032, updated#3033, ... 12 more fields]
               +- Project [_c0#3010, appId#3011, developer#3012, developerId#3013, developerWebsite#3014, free#3015,
genre#3016, genreId#3017, inAppProductPrice#3018, minInstalls#3019, offersIAP#3020, originalPrice#3021, price#3022,
ratings#3023, len screenshots#3024, adSupported#3025, containsAds#3026, reviews#3027, releasedDayYear#3028, sale#3029,
score#3030, summary#3031, title#3032, updated#3033, ... 12 more fields]
                  +- Project [_c0#3010, appId#3011, developer#3012, developerId#3013, developerWebsite#3014,
free#3015, genre#3016, genreId#3017, inAppProductPrice#3018, minInstalls#3019, offersIAP#3020, originalPrice#3021,
price#3022, ratings#3023, len screenshots#3024, adSupported#3025, containsAds#3026, reviews#3027,
releasedDayYear#3028, sale#3029, score#3030, summary#3031, title#3032, updated#3033, ... 12 more fields]
                     +- Project [_c0#3010, appId#3011, developer#3012, developerId#3013, developerWebsite#3014,
free#3015, genre#3016, genreId#3017, inAppProductPrice#3018, minInstalls#3019, offersIAP#3020, originalPrice#3021,
price#3022, ratings#3023, len screenshots#3024, adSupported#3025, containsAds#3026, reviews#3027,
releasedDayYear#3028, sale#3029, score#3030, summary#3031, title#3032, updated#3033, ... 12 more fields]
                        +- Relation
[_c0#3010,appId#3011,developer#3012,developerId#3013,developerWebsite#3014,free#3015,genre#3016,genreId#3017,inAppProdu
screenshots#3024,adSupported#3025,containsAds#3026,reviews#3027,releasedDayYear#3028,sale#3029,score#3030,summary#3031,
12 more fields] csv
```

```
final_output_df.coalesce(1).write.mode("overwrite").option('header',True).option('sep'=";").format('csv').save('dbfs:/FileStore/google_play_dataset_by_tapivedotcom.csv/')
```

Start coding or generate with AI.