

Name:Amruta Naik

NUID:002891105

Individual Project Part 1

ER STUDIO:

Filename:ER-STUDIO: ProjectIndividualAmruta



ProjectIndividualAmru
ta.DM1

SQLscripts ran to create tables in snowflake



DimensionalModellin
gfile.sql

Screen of tables Created on Snowflake:

The screenshot displays the Snowflake web interface. The left sidebar shows a tree view of databases and schemas, including 'CREATE_DB', 'CREATE_SCHEMA', and 'Tables'. The main panel shows the SQL query executed: 'CREATE TABLE DIM_CRIME(' followed by a comma. The 'Results' tab at the bottom indicates 'Statement executed successfully.' and shows 'Query duration: 82ms' and 'Rows: 1'. The 'Query ID' is '01ba6f...' and there is an 'Ask Copilot' button.

Alteryx Profiling

Alteryx is used for profiling and also cleaning the data



finalprojectalteryx.yx
md

Problems identified:

1. Used basic profiling and field summary to identify the issues in the NYPD data arrest file.
2. Identified the Issues and then cleaned the data using Alteryx.

List of issues identified while profiling

- 1.Used select in Alteryx to Change all the data types as per the meta data for all 22 columns.
- 2.Changed the ARREST_DATE to appropriate snowflake acceptable format YYYY-MM-DD saved this with DateTimeNow
- 3.Identified all the null values and replaced all the Numeric values with 0 and string valued are replaced by known Bu using functions in ALTERYX

Identified nulls in

- PD_CD numeric
- PD_DESC string
- KY_CD numeric
- OFNS_DESC string
- LAW_CODE string
- LAW_CAT_CD had 9,I and null values which was handled

4.For latitude longitude and X-coord and Y -coord

Where ever there is 0 or null values. I calculated the mean and then replaced 0 or null values with the Mean for all of the 4 columns.

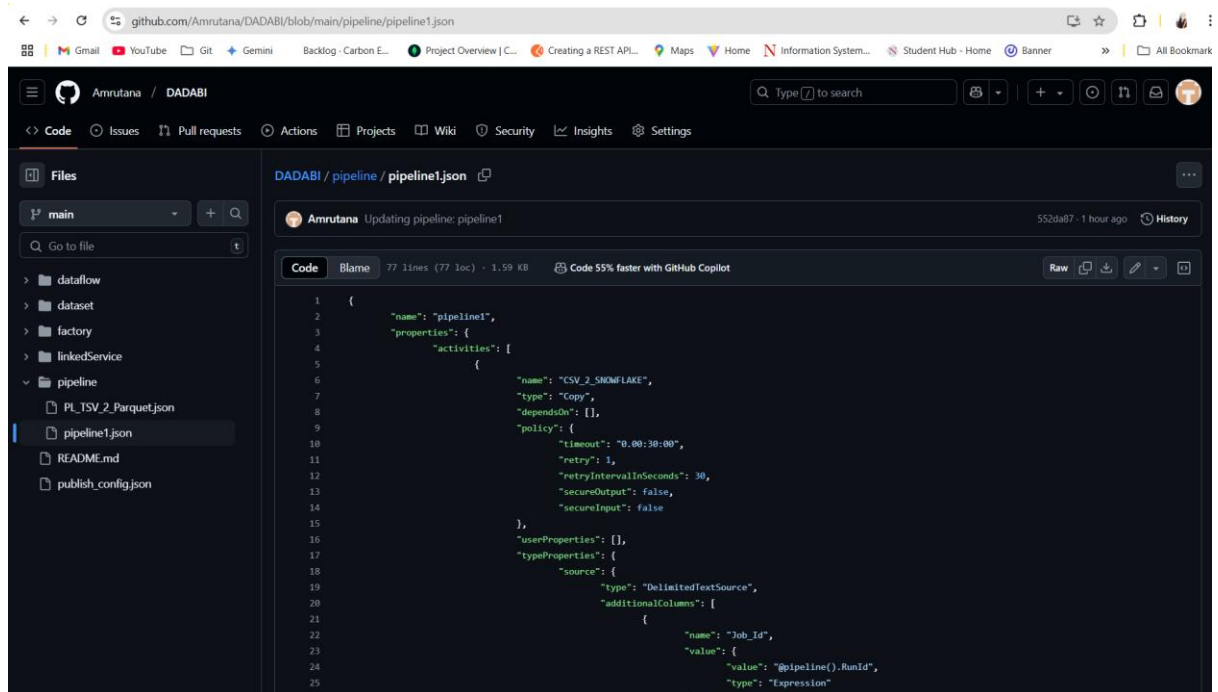
Once profiling is done I stored my data locally and then used CSV to snowflake pipelining and did a stage table in snowflake

Created a stage table in snowflake with

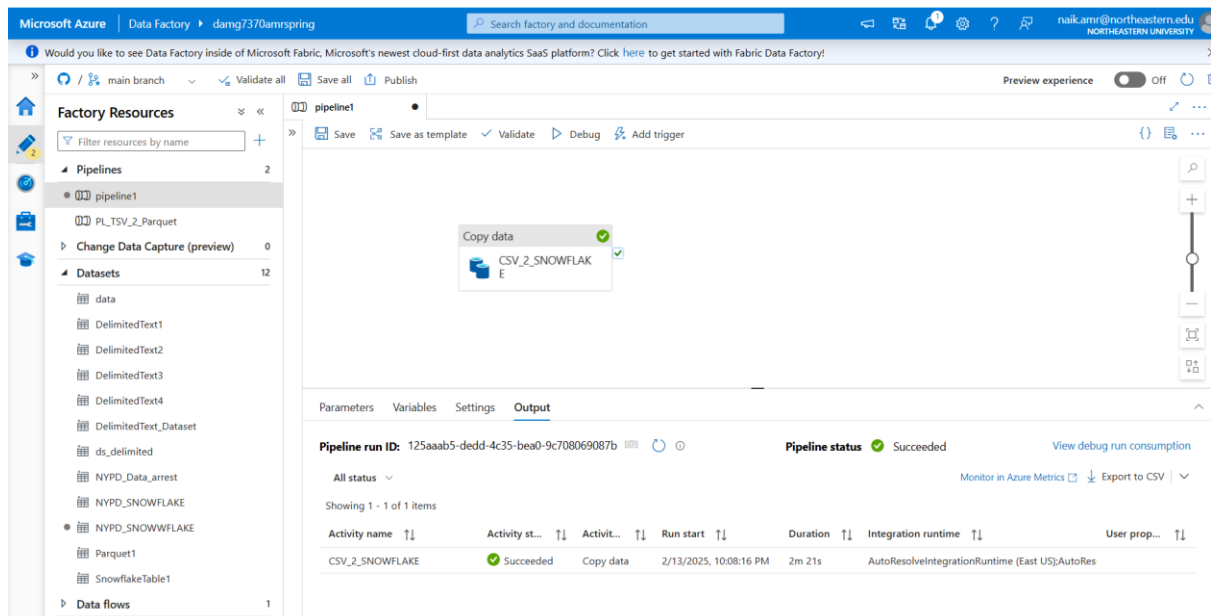
```
CREATE OR REPLACE TABLE NYPD_ARREST_SCHEMA.arrest_stg (  
  ARREST_KEY VARCHAR(16384),  
  ARREST_DATE VARCHAR(16384),  
  PD_CD INT,  
  PD_DESC VARCHAR(16384),  
  KY_CD INT,  
  OFNS_DESC VARCHAR(16384),  
  LAW_CODE VARCHAR(16384),  
  LAW_CAT_CD VARCHAR(16384),  
  ARREST_BORO VARCHAR(16384),  
  ARREST_PRECINCT INT,  
  JURISDICTION_CODE BIGINT,  
  AGE_GROUP VARCHAR(16384),  
  PERP_SEX VARCHAR(16384),  
  PERP_RACE VARCHAR(16384),  
  X_COORD_CD DOUBLE,  
  Y_COORD_CD DOUBLE,  
  Latitude DOUBLE,  
  Longitude DOUBLE,  
  NEW_GEOREFERENCED VARCHAR(16384),  
  DateTime_Out DATE,  
  LOAD_DATE DATE,  
  JOB_ID VARCHAR(50)  
);
```

GIT repository Link

<https://github.com/Amrutana/DADABI/tree/main/pipeline>



ADF pipeline Screenshot:



↖ ↗

Activity run id: f91494c2-019a-453b-b39a-516d002df8dd



- ✓ Azure Data Lake Storage Gen2 → Azure Blob Storage

Used DIUs	4
-----------	---

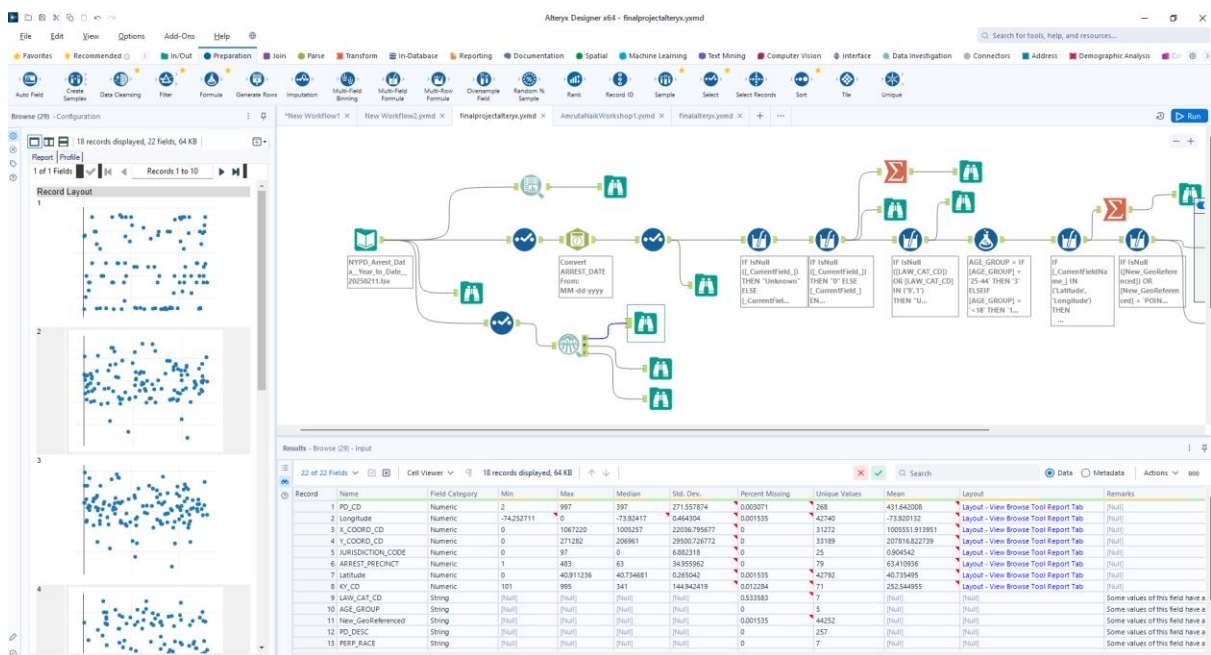
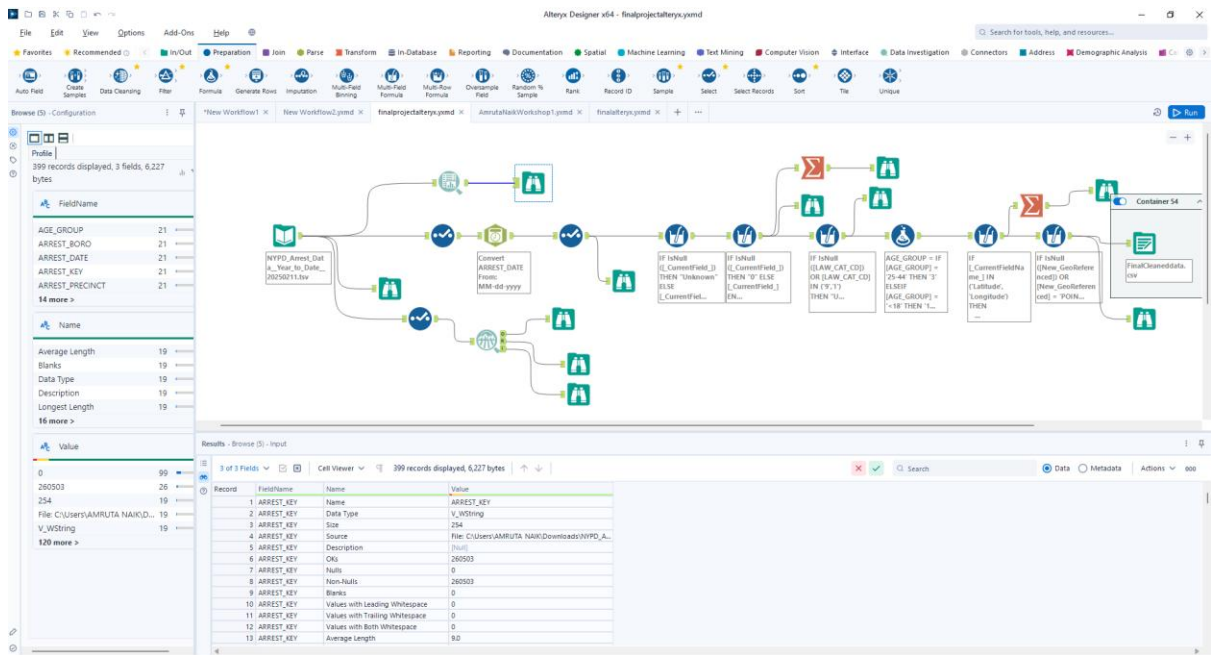
Details

Listing source ⓘ 00:00:00

[illegible]

★★★★★

The screenshot displays the Alteryx Designer x64 interface. The top menu bar includes File, Edit, View, Options, Add-Ons, and Help. Below the menu is a toolbar with icons for Favorites, Recommended, In/Out, Preparation, Join, Parse, Transform, In-Database, Reporting, Documentation, Spatial, Machine Learning, Text Mining, Computer Vision, Interface, Data Investigation, Connectors, Address, Demographic Analysis, and other tools. The main workspace shows a workflow named 'finalprojectalteryx.ymd' with several steps: 'New Field', 'Create Sample', 'Data Cleansing', 'Filter', 'Formula', 'Generate Rows', 'Imputation', 'Multi-Field Formula', 'Multi-Field Formula', 'Multi-Field Formula', 'Overample', 'Random % Sample', 'Rank', 'Record ID', 'Sample', 'Select', 'Select Records', 'Sort', 'Tile', and 'Unique'. The workflow is connected to a 'Container 54' which contains a 'FinalCleanedData.yml' file. A status bar at the bottom indicates 'Finished running finalprojectalteryx.ymd in 23.0 seconds with 10 field conversion errors and 2 warnings using AMP engine'. The bottom panel shows the 'Results - Workflow - Messages' log, detailing the execution of various steps and the final completion message.



Output csv file of cleaned data



FinalCleaneddata.csv

Data Populated in Snowflake after running the pipeline

- Row counts using SQL query

The screenshot shows the Snowflake web interface. On the left, the 'Databases' sidebar is open, showing a tree structure with 'CREATE_DB' and 'CREATE_SCHEMA' folders. Under 'CREATE_SCHEMA', there are several tables listed, including 'NYPD_DATA_ARREST_STG'. The main panel displays a SQL query in the 'CREATE_DB.CREATE_SCHEMA' context. The query is as follows:

```
CREATE OR REPLACE TABLE
  9  LAW_WAVE VARCHAR(16384),
 10  LAW_CAT_CD VARCHAR(16384),
 11  ARREST_BORO VARCHAR(16384),
 12  ARREST_PRECINCT BIGINT,
 13  JURISDICTION_CODE BIGINT,
 14  AGE_GROUP VARCHAR(16384),
 15  PERP_SEX VARCHAR(16384),
 16  PERP_RACE VARCHAR(16384),
 17  X_COORD_CD DOUBLE,
 18  Y_COORD_CD DOUBLE,
 19  Latitude DOUBLE,
 20  Longitude DOUBLE,
 21  NEW_GEOREFERENCED VARCHAR(16384),
 22  DateTime_Out DATE,
 23  Load_Date DATE,
 24  Job_ID VARCHAR(50)
 25 );
 26 select * from NYPD_DATA_ARREST_STG;
 27
 28 select count(*) from NYPD_DATA_ARREST_STG;
```

Below the query, the 'Results' tab is active, showing a single row with the value 260503 for the COUNT(*) query. The 'Query Details' panel on the right shows the query duration as 45ms and the number of rows as 1. The 'NYPD_DATA_ARREST_STG' table is also visible in the sidebar, showing its schema with columns like ARREST_KEY, ARREST_DATE, PD_CD, PD_DESC, KY_CD, OFNS_DESC, LAW_CODE, LAW_CAT_CD, ARREST_BORO, and ARREST_PRECINCT.

Running SQL scripts as per the business requirements

1. How many arrests occurred on any specific day, week, month, quarter, or year?

-- Daily arrests

SELECT

DATE_SY,

COUNT(*) **AS** ArrestCount

FROM FACT_ARRESTS

GROUP BY DATE_SY

ORDER BY DATE_SY;

2025-02-10 9:48pm 2025-02-13 8:27pm 2025-02-13 8:36pm 2025-02-13 8:57pm 2025-02-13 10:16pm 2025-02-13 10:42pm

Databases Workbooks

CREATE_OR_CREATE_SCHEMA Settings

```

1 -- Daily Arrests
2 SELECT
3   DATE_SY,
4   COUNT(*) AS NumberOfArrestsdaily
5 FROM
6   FACT_ARRESTS
7 GROUP BY
8   DATE_SY
9 ORDER BY
10  DATE_SY;
11
12 -- Weekly Arrests
13 SELECT
14   YEAR_NUM,
15   WEEK_NUM,
16   COUNT(*) AS NumberOfArrestsweek
17 FROM
18   FACT_ARRESTS fa
19 JOIN
20   DIMENSION_DATE dd ON fa.DATE_SY = dd.DATE_SY
21 GROUP BY
22   YEAR_NUM,
23   WEEK_NUM
24 ORDER BY
25   YEAR_NUM,

```

Results Chart

DATE_SY	NUMBEROFARRESTSDAILY
Query produced no results	

Query Details

- Query duration: 427ms
- Rows: 0
- Query ID: 03ba1098-9001-30ff-00...

Ask Copilot

-- Weekly arrests

SELECT

YEAR(DATE_SY) **AS** ArrestYear,

DATEPART(wk, DATE_SY) **AS** ArrestWeek,

COUNT(*) **AS** ArrestCount

FROM FACT_ARRESTS

GROUP BY YEAR(DATE_SY), DATEPART(wk, DATE_SY)

ORDER BY YEAR(DATE_SY), DATEPART(wk, DATE_SY);

2025-02-10 9:48pm 2025-02-13 8:27pm 2025-02-13 8:36pm 2025-02-13 8:57pm 2025-02-13 10:16pm 2025-02-13 10:42pm

Databases Workbooks

CREATE_OR_CREATE_SCHEMA Settings

```

7 GROUP BY
8   DATE_SY
9 ORDER BY
10  DATE_SY;
11
12 -- Weekly Arrests
13 SELECT
14   YEAR_NUM,
15   WEEK_NUM,
16   COUNT(*) AS NumberOfArrestsweek
17 FROM
18   FACT_ARRESTS fa
19 JOIN
20   DIMENSION_DATE dd ON fa.DATE_SY = dd.DATE_SY
21 GROUP BY
22   YEAR_NUM,
23   WEEK_NUM
24 ORDER BY
25   YEAR_NUM,
26   WEEK_NUM;
27
28 -- Monthly Arrests
29 SELECT
30   YEAR_NUM,
31   MONTH_NUM,

```

Results Chart

YEAR_NUM	WEEK_NUM	NUMBEROFARRESTSWEEK
Query produced no results		

Query Details

- Query duration: 607ms
- Rows: 0
- Query ID: 03ba1098-9001-3917-0...

Ask Copilot

-- Monthly arrests

SELECT

YEAR(DATE_SY) **AS** ArrestYear,

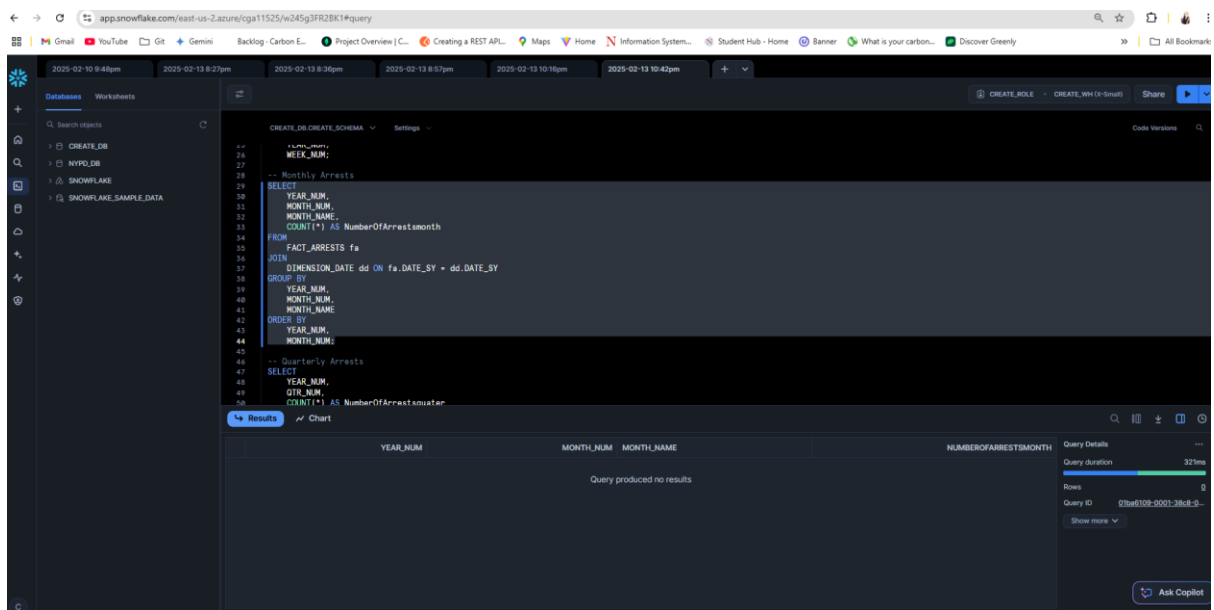
MONTH(DATE_SY) **AS** ArrestMonth,

COUNT(*) **AS** ArrestCount

FROM FACT_ARRESTS

GROUP BY YEAR(DATE_SY), MONTH(DATE_SY)

ORDER BY YEAR(DATE_SY), MONTH(DATE_SY);



-- Quarterly arrests

SELECT

YEAR(DATE_SY) **AS** ArrestYear,

DATEPART(qq, DATE_SY) **AS** ArrestQuarter,

COUNT(*) **AS** ArrestCount

FROM FACT_ARRESTS

GROUP BY YEAR(DATE_SY), DATEPART(qq, DATE_SY)

ORDER BY YEAR(DATE_SY), DATEPART(qq, DATE_SY);

The screenshot shows the Snowflake web interface with a SQL query for quarterly arrests. The query is as follows:

```

45  YEAR_NUM,
46  MONTH_NUM,
47  -- Quarterly Arrests
48  SELECT
49    YEAR_NUM,
50    QTR_NUM,
51    COUNT(*) AS NumberOfArrestsQuarter
52  FROM
53    FACT_ARRESTS fa
54  JOIN
55    DIMENSION_DATE dd ON fa.DATE_SY = dd.DATE_SY
56  GROUP BY
57    YEAR_NUM,
58    QTR_NUM
59  ORDER BY
60    YEAR_NUM,
61    QTR_NUM;
62
63  -- Yearly Arrests
64  SELECT
65    YEAR_NUM,
66    COUNT(*) AS NumberOfArrestsYearly
67  FROM
68    FACT_ARRESTS fa
  
```

The results table is empty, showing columns: YEAR_NUM, QTR_NUM, and NUMBEROFARRESTSQUARTER. The query duration is 520ms.

-- Yearly arrests

SELECT

YEAR(DATE_SY) AS ArrestYear,

COUNT(*) AS ArrestCount

FROM FACT_ARRESTS

GROUP BY YEAR(DATE_SY)

ORDER BY YEAR(DATE_SY);

The screenshot shows the Snowflake web interface with a SQL query for yearly arrests. The query is as follows:

```

49  QTR_NUM,
50  COUNT(*) AS NumberOfArrestsQuarter
51  FROM
52    FACT_ARRESTS fa
53  JOIN
54    DIMENSION_DATE dd ON fa.DATE_SY = dd.DATE_SY
55  GROUP BY
56    YEAR_NUM,
57    QTR_NUM
58  ORDER BY
59    YEAR_NUM,
60    QTR_NUM;
61
62  -- Yearly Arrests
63  SELECT
64    YEAR_NUM,
65    COUNT(*) AS NumberOfArrestsYearly
66  FROM
67    FACT_ARRESTS fa
68  JOIN
69    DIMENSION_DATE dd ON fa.DATE_SY = dd.DATE_SY
70  GROUP BY
71    YEAR_NUM
72  ORDER BY
73    YEAR_NUM;
  
```

The results table is empty, showing columns: YEAR_NUM and NUMBEROFARRESTSYEARLY. The query duration is 56ms.

2. What are the peak days and months for arrests?

-- Peak Days for Arrests

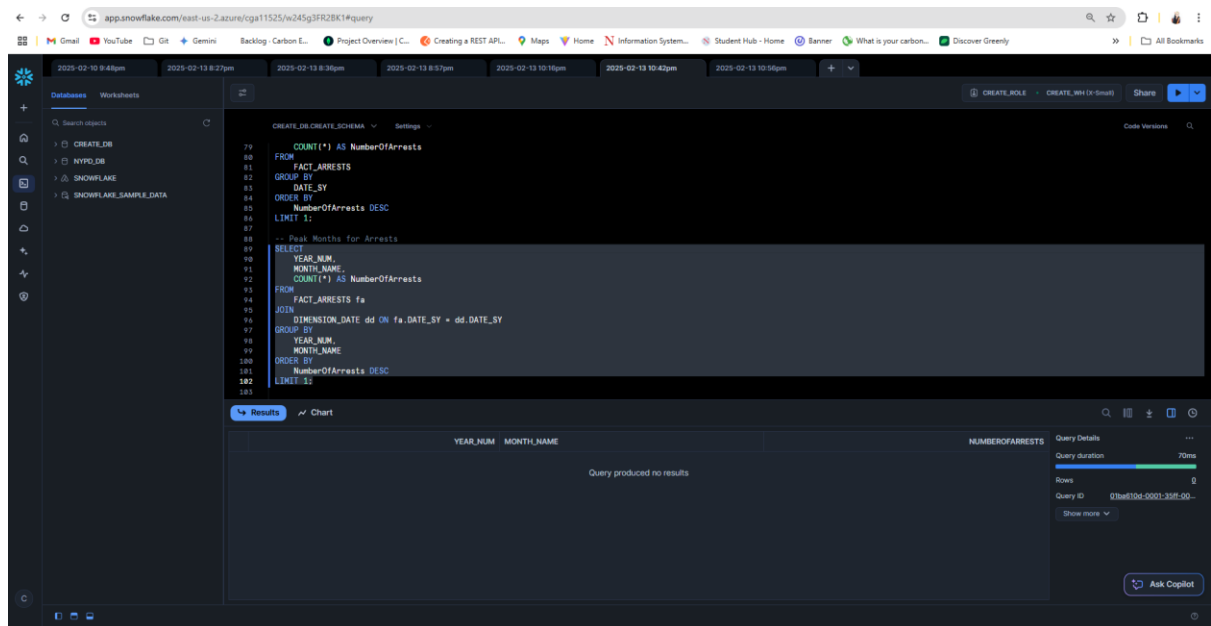
```
SELECT
    DATE_SY,
    COUNT(*) AS NumberOfArrests
FROM
    FACT_ARRESTS
GROUP BY
    DATE_SY
ORDER BY
    NumberOfArrests DESC
LIMIT 1;
```

-- Peak Months for Arrests

```
SELECT
    YEAR_NUM,
    MONTH_NAME,
    COUNT(*) AS NumberOfArrests
FROM
    FACT_ARRESTS fa
JOIN
    DIMENSION_DATE dd ON fa.DATE_SY = dd.DATE_SY
GROUP BY
    YEAR_NUM,
    MONTH_NAME
ORDER BY
    NumberOfArrests DESC
LIMIT 1;
```

The screenshot displays the Snowflake SQL Editor interface. The top section shows a list of tabs for different queries. The active query is the first one, which contains the SQL for finding peak days for arrests. Below the query editor, the 'Results' tab is selected, showing a table with two columns: 'DATE_SY' and 'NUMBEROFARRESTS'. The table is empty, with a message stating 'Query produced no results'. The 'Query Details' panel on the right shows a query duration of 68ms, 5 rows, and a query ID of 01bad10d-5001-37fa-0-.

```
CREATE OR REPLACE SCHEMA
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
```

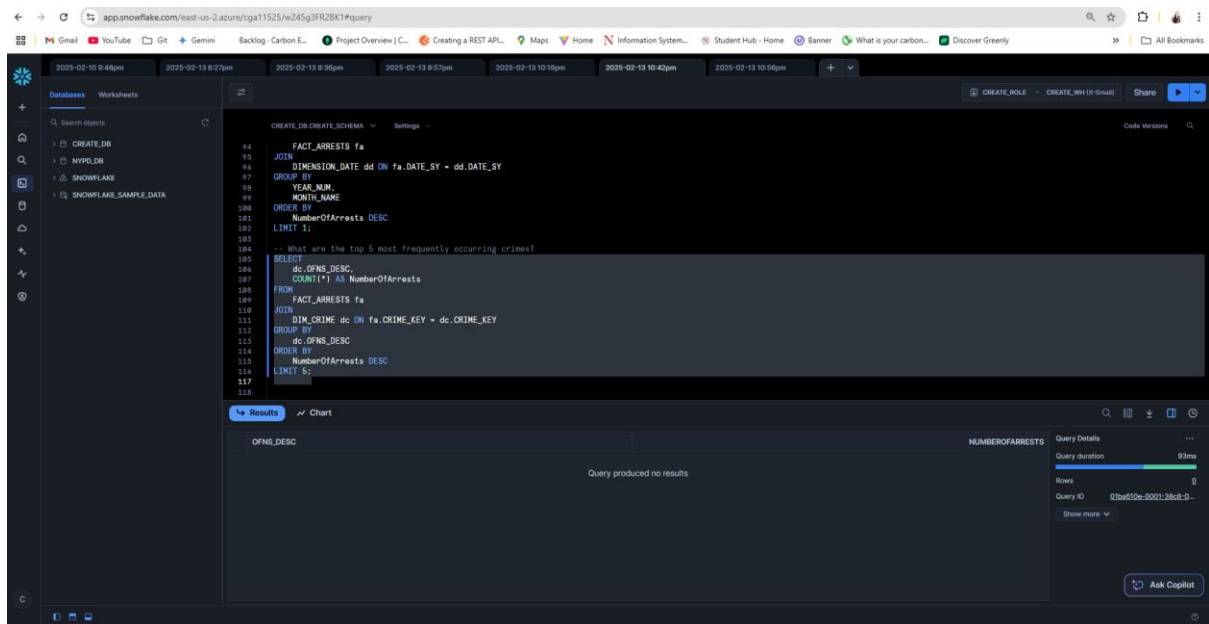


3. What are the top 5 most frequently occurring crimes?

```

SELECT
    dc.OFNS_DESC,
    COUNT(*) AS NumberOfArrests
FROM
    FACT_ARRESTS fa
JOIN
    DIM_CRIME dc ON fa.CRIME_KEY = dc.CRIME_KEY
GROUP BY
    dc.OFNS_DESC
ORDER BY
    NumberOfArrests DESC
LIMIT 5;

```



4. Which crimes have increased or decreased the most over time?

SELECT

dd.YEAR_NUM,

dc.OFNS_DESC,

COUNT(*) AS NumberOfArrests

FROM

FACT_ARRESTS fa

JOIN

DIM_CRIME dc ON fa.CRIME_KEY = dc.CRIME_KEY

JOIN

DIMENSION_DATE dd ON fa.DATE_SY = dd.DATE_SY

GROUP BY

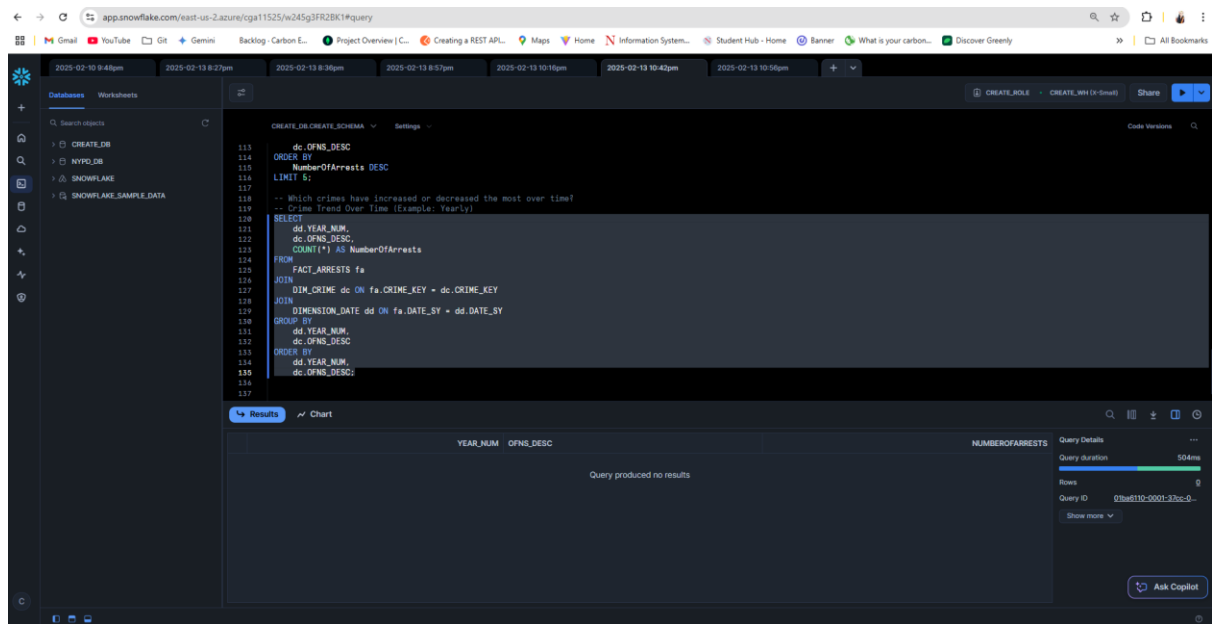
dd.YEAR_NUM,

dc.OFNS_DESC

ORDER BY

dd.YEAR_NUM,

dc.OFNS_DESC;



5. Are there specific precincts with higher felony arrests compared to misdemeanors?

-- Are there specific precincts with higher felony arrests compared to misdemeanors?

SELECT

dl.ARREST_PRECINCT,

dc.LAW_CAT_CD,

COUNT(*) AS NumberOfArrests

FROM

FACT_ARRESTS fa

JOIN

DIMENSION_LOCATION dl ON fa.LOCATION_KEY = dl.LOCATION_KEY

JOIN

DIM_CRIME dc ON fa.CRIME_KEY = dc.CRIME_KEY

WHERE

dc.LAW_CAT_CD IN ('F', 'M') -- Assuming 'F' for Felony, 'M' for Misdemeanor

GROUP BY

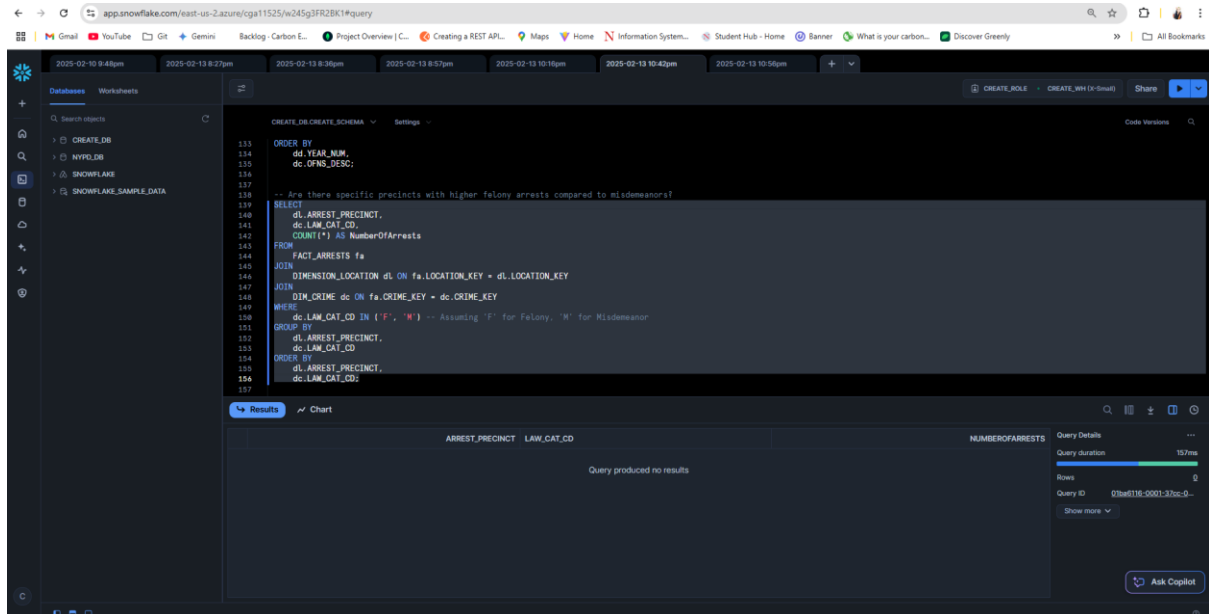
dl.ARREST_PRECINCT,

dc.LAW_CAT_CD

ORDER BY

dl.ARREST_PRECINCT,

dc.LAW_CAT_CD;



6. Which borough has the highest number of arrests?

-- Which borough has the highest number of arrests?

SELECT

dl.BOROUGH_NAME,

COUNT(*) AS NumberOfArrests

FROM

FACT_ARRESTS fa

JOIN

DIMENSION_LOCATION dl ON fa.LOCATION_KEY = dl.LOCATION_KEY

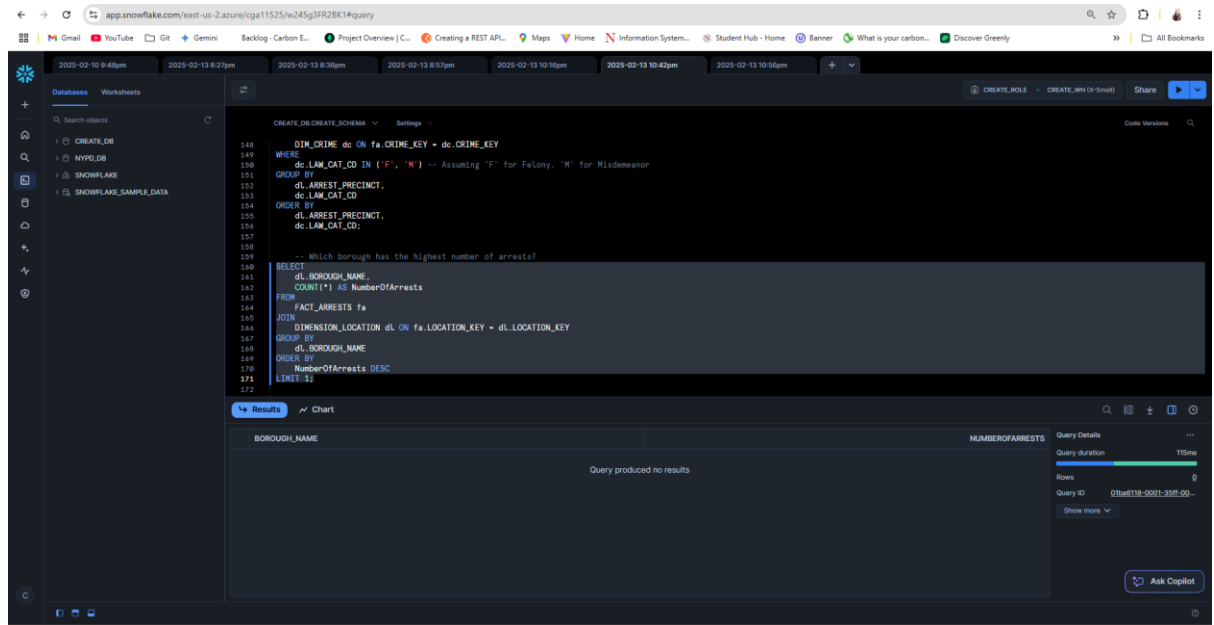
GROUP BY

dl.BOROUGH_NAME

ORDER BY

NumberOfArrests DESC

LIMIT 1;



7. What is the distribution of arrestees by age, race, and gender?

SELECT

dp.AGE_GROUP,
dp.PERP_RACE,
dp.PERP_SEX,
COUNT(*) AS NumberOfArrests

FROM

FACT_ARRESTS fa

JOIN

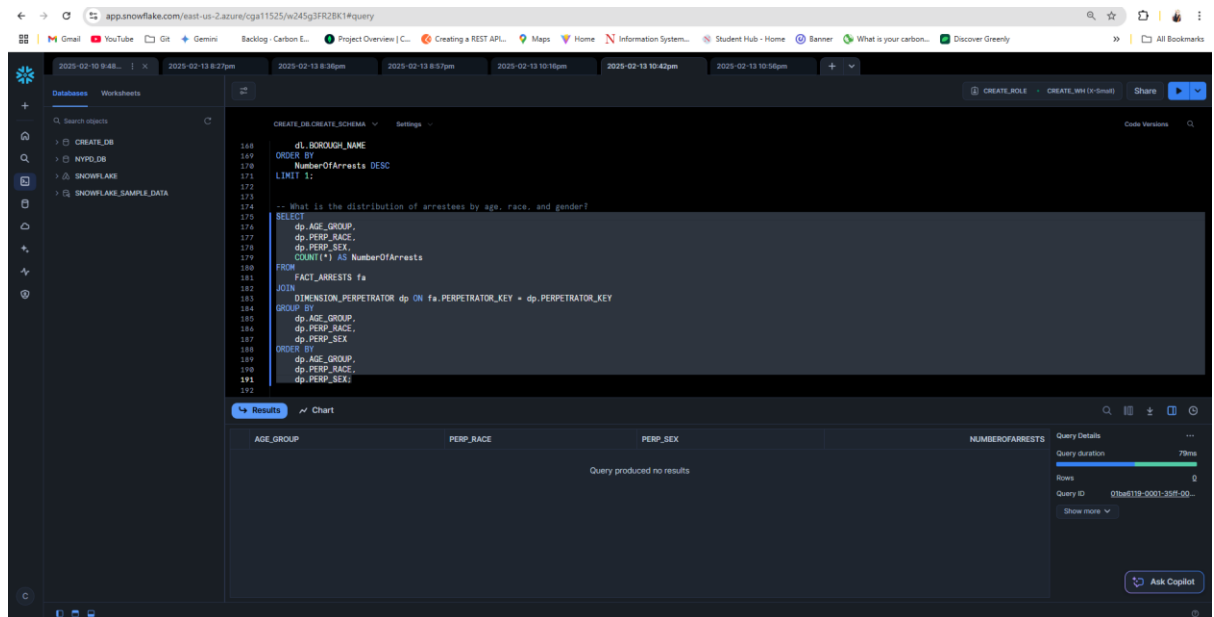
DIMENSION_PERPETRATOR dp ON fa.PERPETRATOR_KEY =
dp.PERPETRATOR_KEY

GROUP BY

dp.AGE_GROUP,
dp.PERP_RACE,
dp.PERP_SEX

ORDER BY

dp.AGE_GROUP,
dp.PERP_RACE,
dp.PERP_SEX;



8. Predict high-crime areas based on past arrest data

SELECT

dl.BOROUGH_NAME,
dl.ARREST_PRECINCT,
dc.CRIME_CATEGORY,
COUNT(*) AS ArrestCount

FROM

FACT_ARRESTS fa

JOIN DIMENSION_LOCATION dl ON fa.LOCATION_KEY = dl.LOCATION_KEY

JOIN DIM_CRIME dc ON fa.CRIME_KEY = dc.CRIME_KEY

GROUP BY dl.BOROUGH_NAME, dl.ARREST_PRECINCT, dc.CRIME_CATEGORY

ORDER BY ArrestCount DESC;

← → ↻

app.snowflake.com/east-us-2.azure/cga11525/w245g3fR2BK1#query

🔍 ⭐ 📌 🖨

Gmail

YouTube

Git

Gemini

Backlog - Carbon E...

Project Overview | C...

Creating a REST APL...

Maps

Home

Information System...

Student Hub - Home

Banner

What is your carbon...

Discover Greenly

39 | All Bookmarks

Databases

Workbooks

Search objects

CREATE_DB

CREATE_SCHEMA

Tables

BIRED_DIM

DATE_DIM

DIMENSION_DATE

DIMENSION_LOCATION

DIMENSION_PERPETRATOR

DIM_CRIME

FACT_ARRESTS

LOCATION_DIM

NYPD_DATA_ARREST_STG

PET_LICENSE_STAGE

PET_LIC_FACT

INFORMATION_SCHEMA

NYPD_DB

SNOWFLAKE

SNOWFLAKE_SAMPLE_DATA

2025-02-10 9:48pm

2025-02-13 8:27pm

2025-02-13 8:36pm

2025-02-13 8:57pm

2025-02-13 10:16pm

2025-02-13 10:43pm

2025-02-13 10:56pm

+

▼

CREATE_DB.CREATE_SCHEMA

Settings

Code Versions

180

181

182

183

184

185

186

187

188

189

190

191

192

193

194

195

196

197

198

199

200

201

202

203

204

```
FROM
FACT_ARRESTS fa
JOIN
  DIMENSION_PERPETRATOR dp ON fa.PERPETRATOR_KEY = dp.PERPETRATOR_KEY
GROUP BY
  dp_AGE_GROUP,
  dp_PERP_RACE,
  dp_PERP_SEX
ORDER BY
  dp_AGE_GROUP,
  dp_PERP_RACE,
  dp_PERP_SEX

SELECT
  dl_BOROUGH_NAME,
  dl_ARREST_PRECINCT,
  dc_CRIME_CATEGORY,
  COUNT(*) AS ArrestCount
FROM
  FACT_ARRESTS fa
JOIN DIMENSION_LOCATION dl ON fa.LOCATION_KEY = dl.LOCATION_KEY
JOIN DIM_CRIME dc ON fa.CRIME_KEY = dc.CRIME_KEY
GROUP BY dl_BOROUGH_NAME, dl_ARREST_PRECINCT, dc_CRIME_CATEGORY
ORDER BY ArrestCount DESC;
```

Results

Chart

BOROUGH_NAME	ARREST_PRECINCT	CRIME_CATEGORY	ARRESTCOUNT
Query produced no results			

Query Details

Query duration 114ms

Rows 8

Query ID 07bad11a-0001-378a-0...

Show more

Ask Copilot