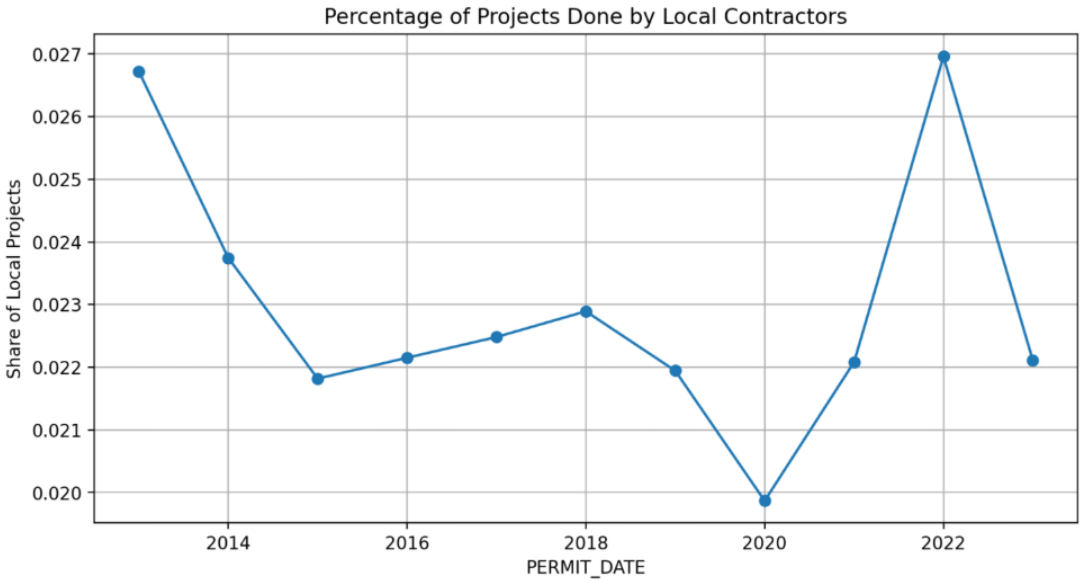
Assume we are **UrbanSC Fix**, a newly established General Contractor, facing a saturated and highly competitive construction market in Los Angeles. Instead of relying on intuition, we utilized public permit data to identify market gaps.

First of all, to determine if we should **focus on localization strategy**, we utilized zip code[1] data in both permit dataset and contractor dataset to analyze. The data reveals a shocking reality: **Less than 3%** of projects are performed by contractors within the same Zip Code. The trend line hovers between **0.020** and **0.027**. This means **97% of contractors** are traveling significant distances to work.



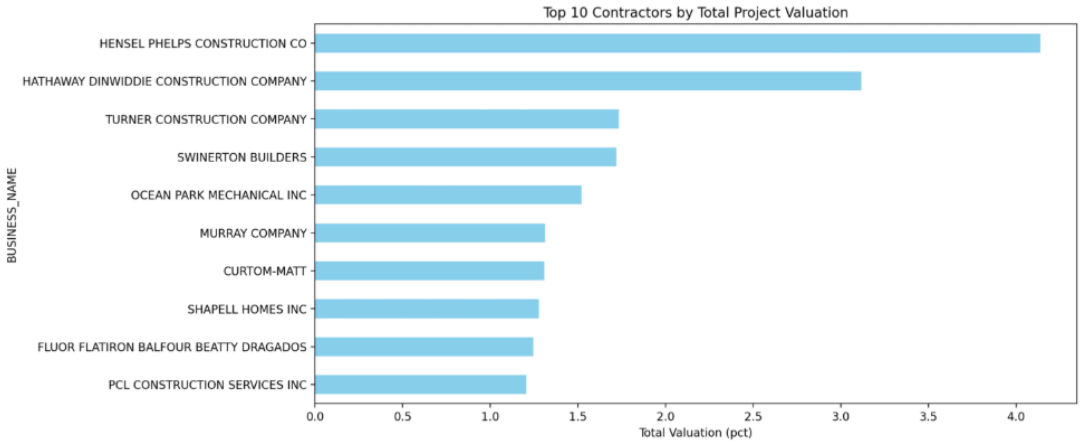

```
#1. Who's using local companies(same zip code means local)
contractor_permits["IS_LOCAL"] = (
    contractor_permits["PROJECT_ZIP"].fillna(0) ==
    contractor_permits["COMPANY_ZIP"].fillna(1)
)

#proportion
local_trends = contractor_permits.groupby(contractor_permits["PERMIT_DATE"].dt.year)["IS_LOCAL"].mean()

#plot
plt.figure(figsize=(10, 5))
local_trends.plot(kind="line", marker="o")
plt.title("Percentage of Projects Done by Local Contractors")
plt.ylabel("Share of Local Projects")
plt.grid(True)
plt.show()
```

We initially thought sticking to our neighborhood was "standard practice." The data proves it is actually a **Blue Ocean Strategy.** Most competitors are **stuck in traffic, driving up costs and delaying timelines**. By strictly enforcing a "Local-Only" service radius, we could become the only contractor who can guarantee **"On-Site in 15 Minutes."** We aren't just "another local builder"; we are a logistical anomaly in a market of commuters. **We will market this as "Zero-Commute Efficiency"—passing the fuel and time savings directly to the customer.**

Then, we turn to analyze our competitors. By computing the total project value that **Top 10** Contractors got, we found that the top players are massive commercial firms like **Hensel**

[1] Local project means zip code between permit records and contractors keeps the same

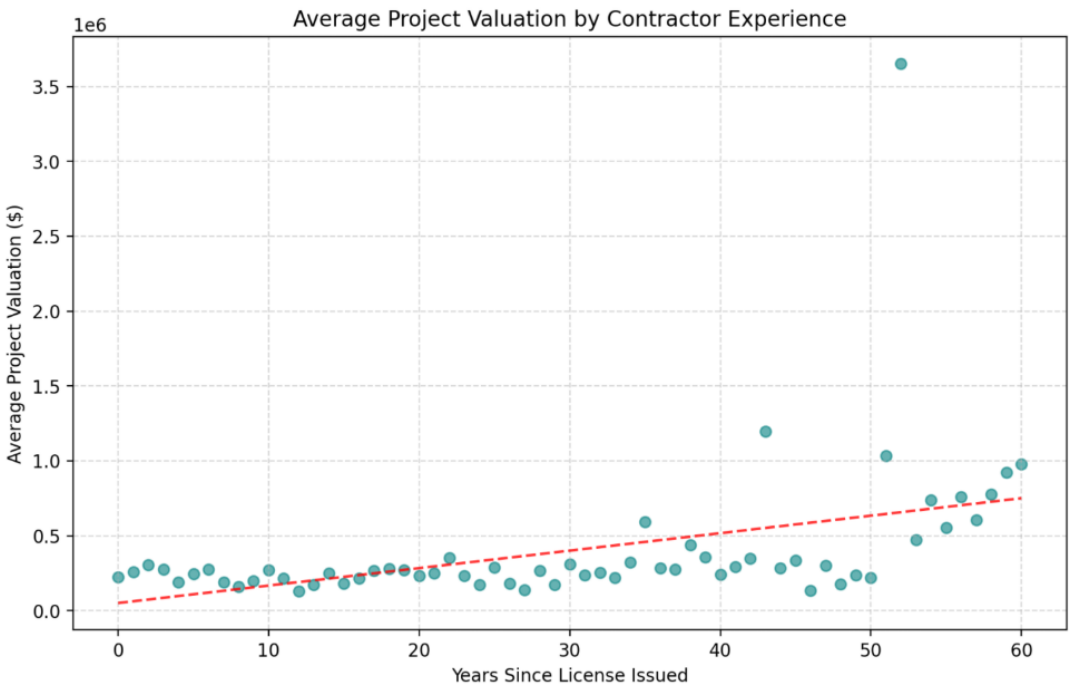
Phelps (4%+) and **Hathaway Dinwiddie**. These companies dominate the high-valuation projects.




```
#2. Are top companies gaining most of the revenue?

#Market share by company
total_market_value = contractor_permits["VALUATION"].sum()

top_builders = (
    contractor_permits.groupby("BUSINESS_NAME")["VALUATION"]
    .sum()
    .sort_values(ascending=False)
    .head(10)
)

top_builders_mkt_share = (top_builders / total_market_value) * 100

#plot
plt.figure(figsize=(12, 6))
top_builders_mkt_share.sort_values().plot(kind="barh", color="skyblue")
plt.title("Top 10 Contractors by Total Project Valuation")
plt.xlabel("Total Valuation (pct)")
plt.show()
```

Though we cannot compete on scale while these top 10 companies hold **nearly 20%** of the market value, **we will focus on the fragment of the market they ignore: individual homeowners (while they prefer to do skyscrapers and stadiums).**

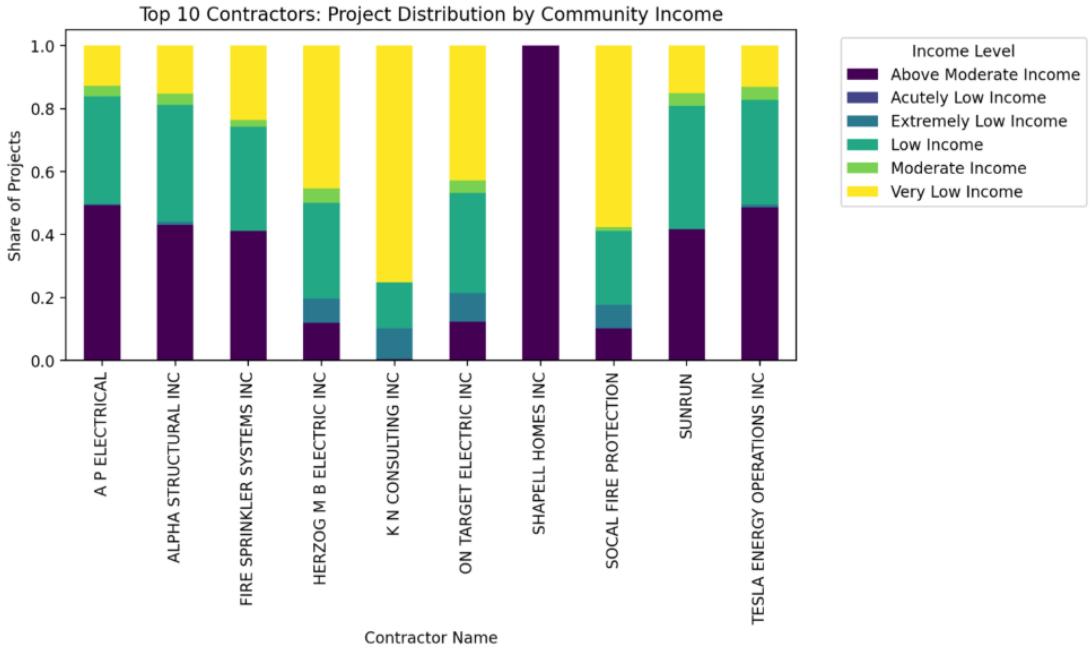
This is not enough since we will not understand the market deeply only by the simple market share analysis. Thus we did another analysis exploring that "**Do established companies have larger project size?**". According to our analysis, the scatter plot shows a flat trend for **the first 10-20 years**. New companies (**License Age < 5**) typically handle projects valued around **$250,000**. It takes **50+ years** of experience to consistently land **$1M+** average valuations.




```python
#3. Do established companies have larger project size

#convert establish date to datetime
contractor_permits["ESTABLISH_DATE"] = pd.to_datetime(contractor_permits["ESTABLISH_DATE"], errors="coerce")

#license age
contractor_permits["LICENSE_AGE"] = (
    contractor_permits["PERMIT_DATE"].dt.year - contractor_permits["ESTABLISH_DATE"].dt.year)

#filter negative and over-60-year values
valid_age = contractor_permits[
    (contractor_permits["LICENSE_AGE"] >= 0) &
    (contractor_permits["LICENSE_AGE"] <= 60)]


age_valuation = valid_age.groupby("LICENSE_AGE")["VALUATION"].mean()

#plot
plt.figure(figsize=(10, 6))
plt.scatter(age_valuation.index, age_valuation.values, color="teal", alpha=0.6)
#Trend line
z = np.polyfit(age_valuation.index, age_valuation.values, 1)
p = np.poly1d(z)
plt.plot(age_valuation.index, p(age_valuation.index), "r--", alpha=0.8)
plt.title("Average Project Valuation by Contractor Experience")
plt.xlabel("Years Since License Issued")
plt.ylabel("Average Project Valuation ($)")
plt.grid(True, linestyle='--', alpha=0.5)
plt.show()
```

Since we are a startup, targeting luxury mansions (**$1M+**) is statistically unrealistic and likely to fail. The strategy is we will target **"Mid-Scale Renovations"** (**$50k - $300k range**). This is the "sweet spot" for new entrants. **We will build our reputation here for the first 5 years before attempting to move up-market.**

Having regional and competitor analyses, we now need some target customer analysis. By connecting income data with top contractor data, we fould that the market shows distinct polartization.

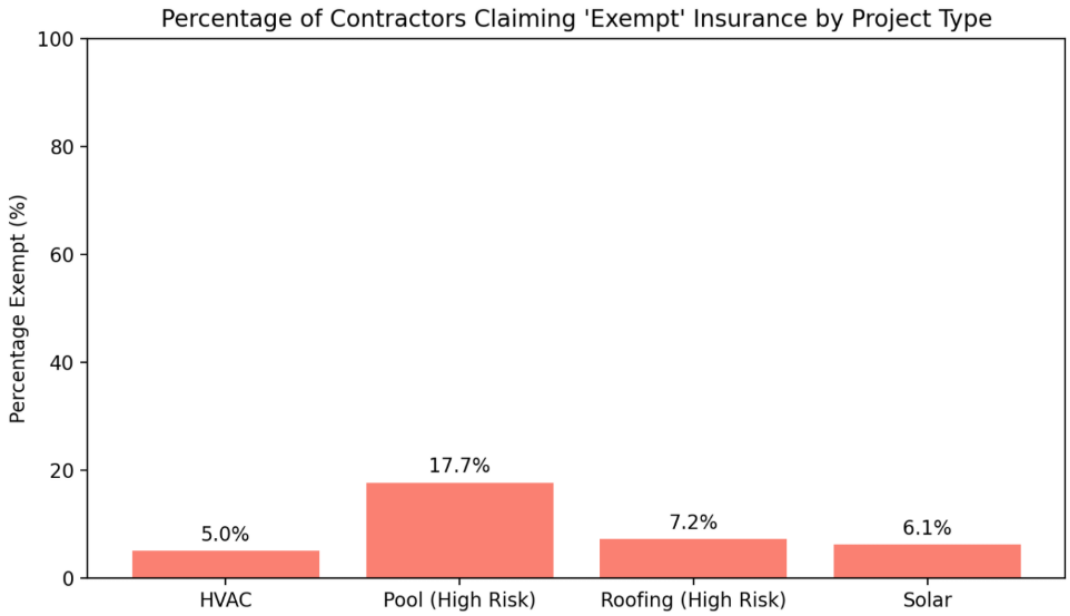



```
#4. Regional preference: Rich community vs Normal community

merged_census = contractor_permits.merge(
    census_tracts,
    left_on="CT",
    right_on="CENSUS_TRACT",
    how="inner"
)

#Top 10 busy companies
top_10_names = merged_census["BUSINESS_NAME"].value_counts().head(10).index.tolist()

subset = merged_census[merged_census["BUSINESS_NAME"].isin(top_10_names)]

#Crosstab
cross_tab = pd.crosstab(subset["BUSINESS_NAME"], subset["AMI_CATEGORY"], normalize="index")

ax = cross_tab.plot(kind="bar", stacked=True, figsize=(10, 6), colormap="viridis")

plt.title("Top 10 Contractors: Project Distribution by Community Income")
plt.ylabel("Share of Projects")
plt.xlabel("Contractor Name")
plt.legend(title="Income Level", bbox_to_anchor=(1.05, 1), loc='upper left')
plt.tight_layout()
plt.show()
```

Companies like Shapell Homes Inc are **almost 100%** focused on **"Above Moderate Income"** customers. Meanwhile, companies like K N Consulting serve a mix of **"Very Low"** and **"Low"** income areas. In fact, The **"Above Moderate"** sector is crowded with specialists like Shapell. The **"Very Low"** sector often lacks budget. **BUT** the "Moderate Income" group are present but not dominated by any single monopoly. Therefore we could **identify out target audience**, which is **the Middle Class**—the neglected center. These clients are too small for Shapell but value quality more than the budget providers.

Above are a couple of basic market analysis, we still get time to go deeper and wider in the future. However, the workers' insurance coverage data seems to be a little bit important—it's about compliance. **According to our market experience, the cost of employee insurance**

is nonnegligible, which may lead to cost saving and compliance risk. By classifying the types of project, we found that the **exempt rate of insurance varies from project type.**




```python
#5. Compliance: did some of companies really paid for worker injury insurance

#identify the risk level of project
def simple_category(desc):
    d = str(desc).lower()
    if "roof" in d: return "Roofing (High Risk)"
    if "pool" in d: return "Pool (High Risk)"
    if "hvac" in d: return "HVAC"
    if "solar" in d: return "Solar"
    return "Other"

contractor_permits["Project_Type"] = contractor_permits["AI_DESCRIPTION"].apply(simple_category)
plot_data = contractor_permits[contractor_permits["Project_Type"] != "Other"].copy()

#
exempt_rates = plot_data.groupby("Project_Type")["WORKERS_COMP_COVERAGE_TYPE"].apply(
    lambda x: (x == "Exempt").mean() * 100
)

#plot
plt.figure(figsize=(9, 5))
bars = plt.bar(exempt_rates.index, exempt_rates.values, color="salmon")
plt.bar_label(bars, fmt='%.1f%%', padding=3)
plt.title("Percentage of Contractors Claiming 'Exempt' Insurance by Project Type")
plt.ylabel("Percentage Exempt (%)")
plt.ylim(0, 100)
plt.show()
```

As we can see, while Roofing (7.2%), Solar (6.1%), and HVAC (5.0%) have high compliance rates, **Swimming Pools stand out with a 17.7% Exempt rate.** This is significantly higher than other high-risk[2] categories, demonstrating that **some contractors may avoid insurance illegally to save cost.** Knowing that, when bidding for pool projects, we will simply show this chart to the client and say "Mr. Client, almost 1 in 5 pool contractors in LA claim they have no employees to avoid insurance. If you hire them and an accident happens, **YOU** are liable. **We are part of the safe 80%.**" That's to say **in HVAC and Roofing, we compete on service (because everyone is safe). In Pools, we compete on Risk Mitigation.** This is a proper

[2] High risk means this type of project needs a large number of workers, which may lead to illegal insurance cost saving

differentiation strategy.

In conclusion, by respecting the data, UrbanSC Fix avoids the trap of competing with commercial giants or expecting unrealistic project valuations. We will operate as a hyper-local logistical specialist for the middle-class residential market, using the industry's widespread non-compliance in pool construction as a powerful tool to win trust and contracts.

Below are code for data preparing and cleaning, and streamlit as well.

Python as cell4

```
# Core Python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import streamlit as st

# We can also use Snowpark for our analyses!
from snowflake.snowpark.context import get_active_session
session = get_active_session()
```

+ Python + SQL + Markdown

Python as cell1

```
session.sql("USE ROLE TRAINING_ROLE").collect()
session.sql("USE DATABASE LA_PERMIT_DATA").collect()
session.sql("USE SCHEMA PUBLIC").collect()
```

	status
0	Statement executed successfully.

Python as cell2

```
records        = session.table("PERMIT_RECORDS").to_pandas()
census_tracts  = session.table("CENSUS_TRACTS").to_pandas()
contractors    = session.table("MASTER_LICENSE").to_pandas()
personnel      = session.table("PERSONNEL_DATA").to_pandas()
worker_comp    = session.table("WORKER_COMP").to_pandas()
```

Python as cell9

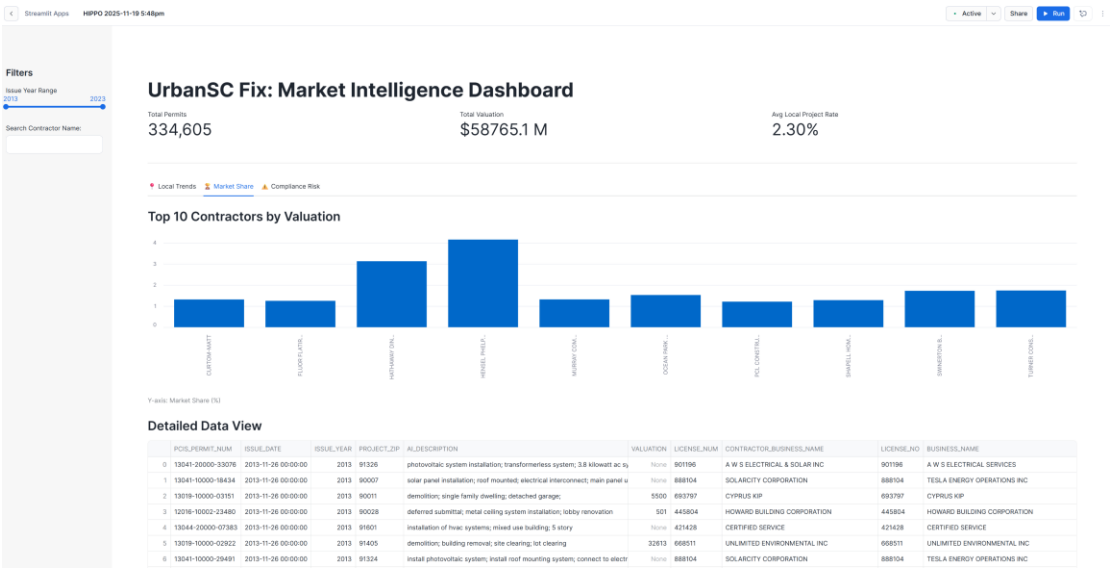
```
#Data clean as we did in class

#Dates
for col in ["ISSUE_DATE", "STATUS_DATE"]:
    records[col] = pd.to_datetime(records[col], errors="coerce")

#Valuation (string -> numeric)
records["VALUATION"] = pd.to_numeric(
        records["VALUATION"].str.replace("$", "").str.replace(",",""),
        errors="coerce")
#Derive CT (LA-specific tract key) from CENSUS_TRACT if present
ct_numeric = pd.to_numeric(records["CENSUS_TRACT"], errors="coerce")
records["CT"] = (ct_numeric * 100) + 6037000000
```

Python as cell6

```
#Clean and Combine datasets
records["LICENSE_NUM"] = records["LICENSE_NUM"].astype(str)
contractors["LICENSE_NO"] = contractors["LICENSE_NO"].astype(str)
contractor_permits = records.merge(contractors,
                                   left_on="LICENSE_NUM",
                                   right_on="LICENSE_NO",
                                   how="inner")
contractor_permits = contractor_permits.rename(columns={"ZIP_CODE_x":"PROJECT_ZIP",
                                                        "ZIP_CODE_y":"COMPANY_ZIP",
                                                       "ISSUE_DATE_x":"PERMIT_DATE",
                                                       "ISSUE_DATE_y":"ESTABLISH_DATE"})

print(len(contractor_permits))
```

```
334605
```

Streamlit:



	PCIS_PERMIT_NUM	ISSUE_DATE	ISSUE_YEAR	PROJECT_ZIP	AI_DESCRIPTION	VALUATION	LICENSE_NUM	CONTRACTOR_BUSINESS_NAME	LICENSE_NO	BUSINESS_NAME
0	13041-20000-33076	2013-11-26 00:00:00	2013	91326	photovoltaic system installation; transformerless system; 3.8 kilowatt ac sy	None	901196	A W S ELECTRICAL & SOLAR INC	901196	A W S ELECTRICAL SERVICES
1	13041-10000-18434	2013-11-26 00:00:00	2013	90007	solar panel installation; roof mounted; electrical interconnect; main panel u	None	888104	SOLARCITY CORPORATION	888104	TESLA ENERGY OPERATIONS INC
2	13019-10000-03151	2013-11-26 00:00:00	2013	90011	demolition; single family dwelling; detached garage;	5500	693797	CYPRUS KIP	693797	CYPRUS KIP
3	12016-10002-23480	2013-11-26 00:00:00	2013	90028	deferred submittal; metal ceiling system installation; lobby renovation	501	445804	HOWARD BUILDING CORPORATION	445804	HOWARD BUILDING CORPORATION
4	13044-20000-07383	2013-11-26 00:00:00	2013	91601	installation of hvac systems; mixed use building; 5 story	None	421428	CERTIFIED SERVICE	421428	CERTIFIED SERVICE
5	13019-10000-02922	2013-11-26 00:00:00	2013	91405	demolition; building removal; site clearing; lot clearing	32613	668511	UNLIMITED ENVIRONMENTAL INC	668511	UNLIMITED ENVIRONMENTAL INC
6	13041-10000-29491	2013-11-26 00:00:00	2013	91324	install photovoltaic system; install roof mounting system; connect to electr	None	888104	SOLARCITY CORPORATION	888104	TESLA ENERGY OPERATIONS INC

```python
# Import python packages
import streamlit as st
import pandas as pd
import numpy as np
from snowflake.snowpark.context import get_active_session

# --- Page Configuration ---
st.set_page_config(layout='wide', page_title="UrbanFix LA Dashboard")
st.title("UrbanSC Fix: Market Intelligence Dashboard")

# Get current Snowflake session
session = get_active_session()

@st.cache_data
def obtain_data():
    # 1. Fetch Permit Records
    permits = session.sql('''
        SELECT pcis_permit_num, issue_date, year(issue_date) as issue_year,
               zip_code as project_zip, ai_description, valuation, license_num,
               contractor_business_name
        FROM la_permit_data.public.permit_records
    ''').to_pandas()

    # 2. Data Cleaning
    # Convert dates
    permits['ISSUE_DATE'] = pd.to_datetime(permits['ISSUE_DATE'])
    # Clean valuation (remove $ and ,) and convert to numeric
    permits['VALUATION'] = pd.to_numeric(
        permits['VALUATION'].astype(str).str.replace(r'[$,]', '', regex=True),
        errors='coerce'
    )
    # Ensure join keys are strings
    permits['LICENSE_NUM'] = permits['LICENSE_NUM'].astype(str)

    # 3. Fetch Master License Data
    # NOTE: Fetching 'workers_comp_coverage_type' directly from here (no separate join needed)
    licenses = session.sql('''
        SELECT license_no, business_name, zip_code as company_zip, workers_comp_coverage_type
        FROM la_permit_data.public.master_license
        WHERE license_no IN (SELECT DISTINCT license_num FROM la_permit_data.public.permit_records)
    ''').to_pandas()
    licenses['LICENSE_NO'] = licenses['LICENSE_NO'].astype(str)

    # 4. Merge Tables
    # Inner join to match permits with contractor details
    df_merged = permits.merge(
        licenses,
        left_on="LICENSE_NUM",
        right_on="LICENSE_NO",
        how="inner"
    )

    # 5. Feature Engineering: Local vs Non-Local
    # Compare Project Zip with Company Zip
    df_merged['IS_LOCAL'] = (
        df_merged['PROJECT_ZIP'].fillna(0) == df_merged['COMPANY_ZIP'].fillna(1)
    )

    return df_merged

# Load Data
df_main = obtain_data()

# --- Sidebar Filters ---
st.sidebar.header('Filters')
```

```python

# 1. Year Slider Filter
if not df_main.empty:
    date0 = int(df_main['ISSUE_YEAR'].min())
    date1 = int(df_main['ISSUE_YEAR'].max())
    min_year, max_year = st.sidebar.slider('Issue Year Range', date0, date1, value=(date0, date1))
else:
    min_year, max_year = 2013, 2024

# 2. Contractor Name Search
contractor_search = st.sidebar.text_input('Search Contractor Name:')

# Function to apply filters
@st.cache_data
def filter_data(df, min_y, max_y, search_term):
    # Filter by Year
    filtered = df[df['ISSUE_YEAR'].between(min_y, max_y)]

    # Filter by Name (if provided)
    if search_term:
        filtered = filtered[filtered['BUSINESS_NAME'].str.contains(search_term, case=False, na=False)]

    return filtered

# Apply the filters
df = filter_data(df_main, min_year, max_year, contractor_search)

# --- Main Metrics Section ---
col1, col2, col3 = st.columns(3)
with col1:
    st.metric('Total Permits', f"{len(df):,}")
with col2:
    # Display in Millions
    st.metric('Total Valuation', f"${df['VALUATION'].sum()/1e6:.1f} M")
with col3:
    # Calculate percentage of local projects
    local_rate = df['IS_LOCAL'].mean() * 100
    st.metric('Avg Local Project Rate', f"{local_rate:.2f}%")

st.divider()

# --- Charts Section (Native Streamlit Charts) ---
tab1, tab2, tab3 = st.tabs(["📍 Local Trends", "⏳ Market Share", "⚠️ Compliance Risk"])

# Chart 1: Localization Trends (Line Chart)
with tab1:
    st.subheader("Contractor Localization Trends")
    # Group by year and calculate mean of boolean IS_LOCAL
    local_trends = df.groupby('ISSUE_YEAR')['IS_LOCAL'].mean()
    st.line_chart(local_trends)
    st.caption("Y-axis: Share of projects done by local contractors (0.02 = 2%)")

# Chart 2: Market Share (Bar Chart)
with tab2:
    st.subheader("Top 10 Contractors by Valuation")
    total_val = df['VALUATION'].sum()

    if total_val > 0:
        # Get Top 10 by Valuation Sum
        top_builders = df.groupby("BUSINESS_NAME")["VALUATION"].sum().sort_values(ascending=False).head(10)
        # Convert to percentage
        top_pct = (top_builders / total_val) * 100
        st.bar_chart(top_pct)
        st.caption("Y-axis: Market Share (%)")
    else:
        st.info("No valuation data available.")

# Chart 3: Compliance Risk (Bar Chart)
with tab3:
    st.subheader("Insurance Exempt Rates by Project Type")

    # Helper function to categorize project types based on description
    def get_category(desc):
        d = str(desc).lower()
        if "roof" in d: return "Roofing (High Risk)"
        if "pool" in d: return "Pool (High Risk)"
        if "hvac" in d: return "HVAC"
        if "solar" in d: return "Solar"
        return "Other"

    # Create a copy to avoid SettingWithCopyWarning
    risk_df = df.copy()
    risk_df["Category"] = risk_df["AI_DESCRIPTION"].apply(get_category)
    plot_data = risk_df[risk_df["Category"] != "Other"]

    wc_col = "WORKERS_COMP_COVERAGE_TYPE"

    if wc_col in plot_data.columns and not plot_data.empty:
        # Calculate percentage of 'Exempt' status
        exempt_rates = plot_data.groupby("Category")[wc_col].apply(lambda x: (x == "Exempt").mean() * 100)
        st.bar_chart(exempt_rates)
        st.caption("Y-axis: Percentage of contractors claiming 'Exempt' status (%)")
    else:
        st.info("No insurance data available in the current selection.")

# --- Detailed Data Table ---
st.subheader('Detailed Data View')
st.dataframe(df.head(100), use_container_width=True)
```