```
import pandas as pd
from datetime import datetime, time

# Load the data
data = pd.read_csv('Kshitiz python project.csv')

# Convert date and time columns
data['currentdate'] = pd.to_datetime(data['currentdate'], format='%m/%d/%Y')
data['currenttime'] = pd.to_datetime(data['currenttime'], format='%H:%M:%S').dt.time
data['datetime'] = pd.to_datetime(data['currentdate'].astype(str) + ' ' +
data['currenttime'].astype(str))

# Sort data by symbol, date, and time
data = data.sort_values(['symbol', 'datetime'])

# Initialize trade log
trade_log = []

def get_first_candle(symbol_data, day):
    """Get the first 5-minute candle of the day."""
    day_data = symbol_data[symbol_data['currentdate'].dt.date == day]
    if not day_data.empty:
        return day_data.iloc[0][['high', 'low', 'close']]
    return None

def calculate_pnl(entry_price, exit_price, trade_type):
    """Calculate profit/loss percentage."""
    if trade_type == 'buy':
        return ((exit_price - entry_price) / entry_price) * 100
    else:  # sell
        return ((entry_price - exit_price) / entry_price) * 100

# Backtest the ORB strategy
symbols = data['symbol'].unique()

for symbol in symbols:
    symbol_data = data[data['symbol'] == symbol]
    days = symbol_data['currentdate'].dt.date.unique()

    for day in days:
        day_data = symbol_data[symbol_data['currentdate'].dt.date == day]
        first_candle = get_first_candle(symbol_data, pd.Timestamp(day))

        if first_candle is None:
            continue

        high_break = first_candle['high']
        low_break = first_candle['low']
```

```
buy_triggered = False
sell_triggered = False
buy_entry_price = None
sell_entry_price = None
buy_entry_time = None
sell_entry_time = None

# Iterate through the day's 5-minute candles
for idx, row in day_data.iterrows():
    current_time = row['currenttime']
    current_price = row['close']
    current_high = row['high']
    current_low = row['low']
    current_datetime = row['datetime']

    # Exit time check (15:15)
    exit_time = time(15, 15)
    if current_time >= exit_time and (buy_triggered or sell_triggered):
        if buy_triggered:
            exit_price = current_price
            pnl = calculate_pnl(buy_entry_price, exit_price, 'buy')
            trade_log.append({
                'date': day,
                'symbol': symbol,
                'trade_type': 'buy',
                'entry_time': buy_entry_time,
                'entry_price': buy_entry_price,
                'exit_time': current_time,
                'exit_price': exit_price,
                'pnl_percent': round(pnl, 2),
                'exit_reason': 'Time Exit (15:15)'
            })
            buy_triggered = False
        if sell_triggered:
            exit_price = current_price
            pnl = calculate_pnl(sell_entry_price, exit_price, 'sell')
            trade_log.append({
                'date': day,
                'symbol': symbol,
                'trade_type': 'sell',
                'entry_time': sell_entry_time,
                'entry_price': sell_entry_price,
                'exit_time': current_time,
                'exit_price': exit_price,
                'pnl_percent': round(pnl, 2),
                'exit_reason': 'Time Exit (15:15)'
            })
            sell_triggered = False
```

```
        continue

    # Buy entry: price crosses above first candle high
    if not buy_triggered and current_high > high_break and not sell_triggered:
        buy_triggered = True
        buy_entry_price = current_price
        buy_entry_time = current_time

    # Sell entry: price crosses below first candle low
    if not sell_triggered and current_low < low_break and not buy_triggered:
        sell_triggered = True
        sell_entry_price = current_price
        sell_entry_time = current_time

    # Check for target or stop loss for buy trade
    if buy_triggered:
        target_price = buy_entry_price * 1.005  # +0.5%
        stop_price = buy_entry_price * 0.9975  # -0.25%

        if current_high >= target_price:
            exit_price = target_price
            pnl = calculate_pnl(buy_entry_price, exit_price, 'buy')
            trade_log.append({
                'date': day,
                'symbol': symbol,
                'trade_type': 'buy',
                'entry_time': buy_entry_time,
                'entry_price': buy_entry_price,
                'exit_time': current_time,
                'exit_price': exit_price,
                'pnl_percent': round(pnl, 2),
                'exit_reason': 'Target Hit'
            })
            buy_triggered = False
        elif current_low <= stop_price:
            exit_price = stop_price
            pnl = calculate_pnl(buy_entry_price, exit_price, 'buy')
            trade_log.append({
                'date': day,
                'symbol': symbol,
                'trade_type': 'buy',
                'entry_time': buy_entry_time,
                'entry_price': buy_entry_price,
                'exit_time': current_time,
                'exit_price': exit_price,
                'pnl_percent': round(pnl, 2),
                'exit_reason': 'Stop Loss Hit'
            })
```

```
                buy_triggered = False

            # Check for target or stop loss for sell trade
            if sell_triggered:
                target_price = sell_entry_price * 0.995  # +0.5% for short
                stop_price = sell_entry_price * 1.0025  # -0.25% for short

                if current_low <= target_price:
                    exit_price = target_price
                    pnl = calculate_pnl(sell_entry_price, exit_price, 'sell')
                    trade_log.append({
                        'date': day,
                        'symbol': symbol,
                        'trade_type': 'sell',
                        'entry_time': sell_entry_time,
                        'entry_price': sell_entry_price,
                        'exit_time': current_time,
                        'exit_price': exit_price,
                        'pnl_percent': round(pnl, 2),
                        'exit_reason': 'Target Hit'
                    })
                    sell_triggered = False
                elif current_high >= stop_price:
                    exit_price = stop_price
                    pnl = calculate_pnl(sell_entry_price, exit_price, 'sell')
                    trade_log.append({
                        'date': day,
                        'symbol': symbol,
                        'trade_type': 'sell',
                        'entry_time': sell_entry_time,
                        'entry_price': sell_entry_price,
                        'exit_time': current_time,
                        'exit_price': exit_price,
                        'pnl_percent': round(pnl, 2),
                        'exit_reason': 'Stop Loss Hit'
                    })
                    sell_triggered = False

# Convert trade log to DataFrame and save to CSV
trade_log_df = pd.DataFrame(trade_log)
trade_log_df.to_csv('trade_log.csv', index=False)
print("Backtest complete. Trade log saved to 'trade_log.csv'.")
print("\nTrade Log:")
print(trade_log_df.to_string(index=False))
```